An Improved Environment for Floats[*]

Anselm Lingnau
`anselm@strathspey.org`

2001/11/08

Abstract

This style option improves the interface for defining floating objects such as figures and tables in LaTeX. It adds the notion of a 'float style' that governs appearance of floats. New kinds of floats may be defined using a `\newfloat` command analogous to `\newtheorem`. This style option also incorporates the functionality of David Carlisle's style option `here`, giving floating environments a `[H]` option which means 'PUT IT HERE' (as opposed to the standard `[h]` option which means 'You may put it here if you like').

1 Introduction

Among the features of LaTeX are 'floating' figures and tables that drift from where they appear in the input text to, say, the top of a page. The contents and formatting of floats is pretty much up to the user, except that there is a `\caption` command that governs formatting of the caption — it is centered if it is short, and formatted as a paragraph if it is longer than a single line of text. Sometimes other types of floating objects, e.g., algorithms or programs, are desirable, but they must be defined by analogy to the existing floats since there is no simple command for doing this. This goes beyond the knowledge or inclination of the average LaTeX user.

In this style option, I present an interface to floating objects that attempts to fix some of these shortcomings. First of all, I introduce the notion of a 'float style'. A float style governs the appearance of a class of floats like a page style governs the appearance of a page (LaTeX has page styles `plain`, `empty` and `headings`, among others). This style option provides some exemplary float styles:

plain This is the float style that LaTeX normally applies to its floats, i.e., nothing in particular. The only difference is that the caption comes out *below* the body of the float, regardless of where it is given in the text.

[*]This file has version number v1.3d. Part of this style option is based on the `here` option by David P. Carlisle (`carlisle@cs.man.ac.uk`), who also provided helpful criticism.

Program 1.1 The first program. This hasn't got anything to do with the package but is included as an example. Note the `ruled` float style.

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int i;
        for (i = 0; i < argc; ++i)
                printf("argv[%d] = %s\n", i, argv[i]);
        return 0;
}
```

plaintop This is similar to 'plain' but the caption always comes out *above* the body of the float.

boxed The body of the float is printed inside a box. The caption goes below that box.

ruled This float style is patterned on the table style of *Concrete Mathematics*. The caption is printed at the top of the float, surrounded by rules; another rule finishes off the float.

To facilitate the definition of new floating objects, `float` supports the `\newfloat` command. This command is comparable to `\newtheorem` in that it allows the user to add a new class of floats at the document level. No style option hacking is necessary. There's also a `\listof` command that prints a listing of all the floats of a given type, like `\listoffigures` and `\listoftables` in vanilla LaTeX.

2 The User Interface — New Floats

`\newfloat` The most important command in `float` is the `\newfloat` command. As mentioned above, it is patterned on `\newtheorem`. The `\newfloat` command takes three required and one optional argument; it is of the form

`\newfloat{`⟨*type*⟩`}{`⟨*placement*⟩`}{`⟨*ext*⟩`}[`⟨*within*⟩`]`

⟨*type*⟩ is the 'type' of the new class of floats, like `program` or `algorithm`. After the appropriate `\newfloat`, commands like `\begin{program}` or `\end{algorithm*}` will be available. ⟨*placement*⟩ gives the default placement parameters for this class of floats. The placement parameters are the same as in standard LaTeX, i.e., `t`, `b`, `p` and `h` for 'top', 'bottom', 'page' and 'here', respectively. When LaTeX writes the captions to an auxiliary file for the list of figures (or whatever), it'll use the job name followed by ⟨*ext*⟩ as a file name. Finally, the optional argument ⟨*within*⟩ determines whether floats of this class will be numbered within some sectional unit of the document. For example, if ⟨*within*⟩=`chapter`, the floats will be numbered within chapters. (In standard LaTeX, this happens with figures and tables in the report and book document styles.) As an example, Program 1.1 above was created by a command sequence similar to that shown in the following Example.

```
\floatstyle{ruled}
\newfloat{Program}{tbp}{lop}[section]
... loads o' stuff ...
\begin{Program}
\begin{verbatim}
... program text ...
\end{verbatim}
\caption{... caption ...}
\end{Program}
```

Example 2.1: This is another silly floating Example. Except that this one doesn't actually float because it uses the `[H]` optional parameter to appear **Here**. (Gotcha.)

`\floatstyle` The `\floatstyle` command sets a default float style. This float style will be used for all the floats that are subsequently defined using `\newfloat`, until another `\floatstyle` command appears. The `\floatstyle` command takes one argument, the name of a float style. For instance, `\floatstyle{ruled}`. Specifying a string that does not name a valid float style is an error.

`\floatname` The `\floatname` command lets you define the *float name* that LaTeX uses in the caption of a float, i.e., 'Figure' for a figure and so on. For example, `\floatname{program}{Program}`. The `\newfloat` command sets the float name to its argument ⟨*type*⟩ if no other name has been specified before.

`\floatplacement` The `\floatplacement` command resets the default placement specifier of a class of floats. E.g., `\floatplacement{figure}{tp}`.

`\restylefloat` The `\restylefloat` command is necessary to change styles for the standard float types `figure` and `table`. Since these aren't usually defined via `\newfloat`, they don't have a style associated with them. Thus you have to say, for example,

```
\floatstyle{ruled}
\restylefloat{table}
```

to have tables come out `ruled`. The command also lets you change style for floats that you define via `\newfloat`, although this is, typographically speaking, not a good idea. See table 1 for an example. There is a `\restylefloat*` command which will restyle an existing float type but will keep the new float style from taking over the `\caption` command. In this case the user is responsible for handling their own captions.

`\listof` The `\listof` command produces a list of all the floats of a given class. Its syntax is

`\listof{`⟨*type*⟩`}{`⟨*title*⟩`}`

⟨*type*⟩ is the float type given in the `\newfloat` command. ⟨*title*⟩ is used for the title of the list as well as the headings if the current page style includes them.

n	$\binom{n}{0}$	$\binom{n}{1}$	$\binom{n}{2}$	$\binom{n}{3}$	$\binom{n}{4}$	$\binom{n}{5}$	$\binom{n}{6}$	$\binom{n}{7}$
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1

Table 1: Pascal's triangle. This is a re-styled LaTeX `table`.

Otherwise, the `\listof` command is analogous to the built-in LaTeX commands `\listoffigures` and `\listoftables`.

3 The User Interface — `[H]` Placement Specifier

Many people find LaTeX's float placement specifiers too restrictive. A Commonly Uttered Complaint (CUC) calls for a way to place a float exactly at the spot where it occurs in the input file, i.e., to *not* have it float at all. It seems that the `[h]` specifier should do that, but in fact it only suggests to LaTeX something along the lines of "put the float here if it's OK with you". As it turns out, LaTeX hardly ever feels inclined to actually do that. This situation can be improved by judicious manipulation of float style parameters.

The same effect can be achieved by changing the actual method of placing floats. David Carlisle's here option introduces a new float placement specifier, namely `[H]`, which, when added to a float, tells LaTeX to "put it HERE, period". If there isn't enough space left on the page, the float is carried over to the next page together with whatever follows, even though there might still be room left for some of that. This style option provides the `[H]` specifier for newly defined classes of floats as well as the predefined `figure`s and `table`s, thereby superseding here. David suggests that the here option be withdrawn from the archives in due course.

The `[H]` specifier may simply be added to the float as an optional argument, like all the other specifiers. It may *not* be used in conjunction with any other placement specifiers, so `[Hhtbp]` is illegal. Neither may it be used as the default placement specifier for a whole class of floats. The following table is defined like this:

```
\begin{table}[H]
\begin{tabular}{cl}
\tt t & Top of the page\\
... more stuff ...
```

(It seems that I have to add some extraneous chatter here just so that the float actually comes out right in the middle of a printed page. When I LaTeXed the documentation just now it turned out that there was a page break that fell exactly between the "So now" line and the float. This wouldn't Prove Anything. Bother.) So now we have the following float placement specifiers:

`t`	Top of the page
`b`	Bottom of the page
`p`	Page of floats
`h`	Here, if possible
`H`	Here, definitely

4 Implementation

4.1 Basics

```
⟨*package⟩
```

In LaTeX, floats are assigned 'type numbers' that are powers of 2. Since there are only two classes of floats, their type numbers are hardwired into the document styles. We need to be somewhat more flexible, and thus we initialize a counter to hold the next type number to be assigned. This counter will be incremented appropriately later.

```
\newcounter{float@type}
\@ifundefined{c@figure}%
  {\setcounter{float@type}{1}}%
  {\setcounter{float@type}{4}}
```

To warm up, we'll look at some of the simpler commands first.

`\floatstyle` The `\floatstyle` command puts its argument into the `\float@style` macro as the name of the new float style. But if the argument doesn't denote a float style, an error message is output instead: Each float style ⟨*style*⟩ has a corresponding command `\fs@`⟨*style*⟩ that contains the appropriate declarations. If the control sequence `\fs@`⟨*arg*⟩ (which goes with the argument ⟨*arg*⟩ to `\floatstyle`) is undefined, i.e., equals `\relax` under `\ifx`, then the float style ⟨*arg*⟩ is unknown, and we call `\float@error{`⟨*arg*⟩`}` for the error message.

```
\newcommand\floatstyle[1]{\@ifundefined{fs@#1}%
  {\float@error{#1}}%
  {\def\float@style{#1}}}
```

`\float@error` Here's the error message. `\@eha` is the help message that says 'Your command was ignored.'

```
\newcommand\float@error[1]{\PackageError{float}{%
  Unknown float style '#1'
}{\@eha}}
```

The next two commands are even simpler. LaTeX says that `\fps@`⟨*float*⟩ contains the default placement specifier for the class of floats ⟨*float*⟩. `\fname@`⟨*float*⟩ expands to the name that appears in ⟨*float*⟩ captions, e.g., 'Figure'. (This is our own definition.)

```
\newcommand\floatname[2]{\@namedef{fname@#1}{#2}}
\newcommand\floatplacement[2]{\@namedef{fps@#1}{#2}}
```

```
\newcommand\floatevery[2]{\csname @float@every@#1\endcsname={#2}}
```

`\restylefloat` The `\restylefloat` command sets up everything so that subsequent commands like `\begin{`⟨*float*⟩`}` use the appropriate float style. It defines `\fst@`⟨*float*⟩ to expand to a command that sets up the currently selected float style (`\fs@`⟨*style*⟩). Then it defines the commands `\begin{`⟨*float*⟩`}`, `\end{`⟨*float*⟩`}`, `\begin{`⟨*float*⟩`*}` and `\end{`⟨*float*⟩`*}`. The `\restylefloat*` command is like `\restylefloat` but will leave the captions of that float type alone.

```
\newcommand\restylefloat{\@ifstar\float@restyle@\float@restyle}
\newcommand\float@restyle@[1]{\float@restyle{#1}%
  \expandafter\let\csname @float@c@#1\endcsname=\@caption}
\newcommand\float@restyle[1]{\expandafter\edef\csname
  fst@#1\endcsname{\expandafter\noexpand\csname
    fs@\float@style\endcsname}%
   \@namedef{#1}{\@nameuse{fst@#1}%
      \@float@setevery{#1}\@float{#1}}%
   \@namedef{#1*}{\@nameuse{fst@#1}%
      \@float@setevery{#1}\@dblfloat{#1}}%
   \expandafter\let\csname end#1\endcsname\float@end
   \expandafter\let\csname end#1*\endcsname\float@dblend
   \expandafter\let\csname @float@c@#1\endcsname=\float@caption
   \@ifundefined{@float@every@#1}{%
     \expandafter\newtoks\csname @float@every@#1\endcsname}{}%
   \@nameuse{@float@every@#1}={}}
```

`\newfloat` Now we can explain how to define a new class of floats. Recall that the three required arguments to `\newfloat` are ⟨*type*⟩, ⟨*placement*⟩ and ⟨*ext*⟩, respectively. First we save the latter two; we also maintain a list of active ⟨*ext*⟩s so we can later iterate over all currently-open lists of floats.

```
\newtoks\float@exts
\newcommand\newfloat[3]{\@namedef{ext@#1}{#3}
  \let\float@do=\relax
  \xdef\@tempa{\noexpand\float@exts{\the\float@exts \float@do{#3}}}%
  \@tempa
  \floatplacement{#1}{#2}%
```

Then we figure out a default value for the 'caption name' of this class of floats. If the `\fname@`⟨*type*⟩ isn't already defined, we tentatively use ⟨*type*⟩ as the name. This is convenient if ⟨*type*⟩ is, say, `Program`, since no `\floatname` command is necessary at all.

```
  \@ifundefined{fname@#1}{\floatname{#1}{#1}}{}
```

Then we set up the type number for LaTeX in `\ftype@`⟨*type*⟩. Afterwards we have to set the `float@type` to the next greater power of two, so that it is ready for the next `\newfloat`. Fortunately, we just have to double it by adding. We don't bother checking for overflow since it is pretty unlikely that somebody will define 25 different classes of floats. Finally, we call `\restylefloat` to define the style and commands for this class of floats.

```
\expandafter\edef\csname ftype@#1\endcsname{\value{float@type}}%
\addtocounter{float@type}{\value{float@type}}
\restylefloat{#1}%
```

Now all that's left is to assemble the `\fnum@`⟨*type*⟩ macro that LaTeX wants to use in its captions. Basically it is just 'caption name' + 'counter value', disguised so that the command *names* appear in the definition instead of their expansions.

```
\expandafter\edef\csname fnum@#1\endcsname%
  {\expandafter\noexpand\csname fname@#1\endcsname{}
     \expandafter\noexpand\csname the#1\endcsname}
```

Finally, we have to take care of the optional argument, ⟨*within*⟩. If the optional argument is present, we pass control to `\float@newx`. Otherwise, we just define the counter for this class of floats. By default, the numbers come out `\arabic`.

```
\@ifnextchar[{\float@newx{#1}}%
  {\@ifundefined{c@#1}{\newcounter{#1}\@namedef{the#1}{\arabic{#1}}}%
    {}}}
```

`\float@newx` Here we deal with the optional argument to `\newfloat`. We have to create a new counter as per `\newcounter{`⟨*type*⟩`}` and add that counter to the list of counters to be reset whenever counter ⟨*within*⟩ is stepped. The standard command `\newcounter{`⟨*type*⟩`}[`⟨*within*⟩`]` takes care of that. However, we can't define the ⟨*type*⟩ counter if it's already defined. While this case is simply ignored when ⟨*within*⟩ is not present, we issue a warning here since what comes out is probably not what the user expects.

```
\def\float@newx#1[#2]{\@ifundefined{c@#1}{\newcounter{#1}[#2]%
    \expandafter\edef\csname the#1\endcsname{%
      \expandafter\noexpand\csname
        the#2\endcsname.\noexpand\arabic{#1}}}%
    {\PackageWarning{float}{Can't redefine counter variable for #1.}}}
```

4.2 Adapting LaTeX internals

We have to adapt some of LaTeX's internal macros to our needs. There are several things that have to be changed around in order to provide the functionality of David Carlisle's here. The following is thus lifted from here, with changes and with David's permission:

`\@float@Hx` We save the original version of `\@xfloat`. (This macro is called from `\@float`, which we used above to define the environment commands for a new class of floats.)

```
\let\@float@Hx\@xfloat
```

`\@xfloat` The new version of `\@xfloat` looks for a `[H]` argument. If it is present `\@float@HH` is called, otherwise the original macro (renamed to `\@float@Hx`) is called.

```
\def\@xfloat#1[{\@ifnextchar{H}{\@float@HH{#1}[}{\@float@Hx{#1}[}}
```

Later on we'll need a box to save a `[H]` float.

```
\newsavebox\float@box
\newif\if@flstyle
```

`\@float@HH` First gobble the `[H]`. Note that `H` should not be used in conjunction with the other placement options, nor as the value of the default placement, as set in `\fps@`*type*.

```
\def\@float@HH#1[H]{%
```

Locally redefine the end of the environment.

```
  \expandafter\let\csname end#1\endcsname\float@endH
```

We don't get a `\@currbox` if we don't actually use the float mechanism. Therefore we fake one using the `\float@box` defined above.

```
  \let\@currbox\float@box
```

Now we save the current float class name for use in constructing the `\caption`. The caption box (defined below) is initialised to an empty box to avoid trouble with floats not having a caption. Then we start the box that'll hold the float itself. `\parindent` is set to zero for compatibility with the standard `[h]` option.

```
  \def\@captype{#1}\setbox\@floatcapt=\vbox{}%
  \expandafter\ifx\csname fst@#1\endcsname\relax
    \@flstylefalse\else\@flstyletrue\fi
  \setbox\@currbox\color@vbox\normalcolor
    \vbox\bgroup \hsize\columnwidth \@parboxrestore
      \@floatboxreset \@setnobreak
```

The final `\ignorespaces` is needed to gobble any spaces or newlines after the `[H]` tokens.

```
  \ignorespaces}
```

```
\newtoks\@float@everytoks
\let\@float@boxreset=\@floatboxreset
\def\@floatboxreset{\@float@boxreset\the\@float@everytoks}
\def\@float@setevery#1{\@float@everytoks=\@nameuse{@float@every@#1}}
```

`\float@makebox` Basically, we must arrange for 'style commands' to be executed at certain points during the generation of the float. LaTeX puts a float into a vertical box `\@currbox` which it takes off a list of empty boxes for insertions. When the `\float@makebox` macro is called, `\@currbox` contains the complete float, minus the caption — we'll see later that we use our own `\caption` command to put the caption into a `\vbox` of its own. This is the only way we can control the position of the caption by the float style, regardless of where the caption appears in the float's input text itself. So the 'style commands' are `\@fs@pre`, which is inserted at the very beginning of the float, `\@fs@mid`, which comes between the float and the caption (or the caption and the float, if captions are put at the top), and `\@fs@post`, which finishes off the float. The caption may appear at the top or at the bottom of the float, as defined by `\@fs@iftopcapt`. Therefore, before we hand the float to LaTeX for positioning, it is taken apart and reassembled using the style commands.

```
\newcommand\float@makebox[1]{%
  \vbox{\hsize=#1 \@parboxrestore
    \@fs@pre\@fs@iftopcapt
      \ifvoid\@floatcapt\else\unvbox\@floatcapt\par\@fs@mid\fi
      \unvbox\@currbox
    \else\unvbox\@currbox
      \ifvoid\@floatcapt\else\par\@fs@mid\unvbox\@floatcapt\fi
    \fi\par\@fs@post\vskip\z@}}
```

`\float@end` The internal macro `\end@float` appears here under the name of `\float@end`. The main thing which is changed is that we call `\float@makebox` to reconstruct the float according to the float style. We want to do exactly what the LaTeX kernel does without copying actual kernel code if we can help it; therefore we finish off the float using the kernel `\@endfloatbox`, then replace LaTeX's contents of the `\@currbox` with our own processed version, and then hand the thing off to LaTeXagain. Of course we have already done `\@endfloatbox`, which comes at the beginning of `\end@float`, ourselves; therefore we neutralize it before calling `\end@float`. This doesn't matter since we're in a group anyway (we wanted to keep the style commands local), so everything is undone at the end of the environment.

```
\newcommand\float@end{\@endfloatbox
  \global\setbox\@currbox\float@makebox\columnwidth
  \let\@endfloatbox\relax\end@float}
```

`\float@endH` The `\float@endH` command is, again, derived from here. It'll deal correctly with a non-floating float, inserting the proper amounts of white space above and below.

```
\newcommand\float@endH{\@endfloatbox\vskip\intextsep
  \if@flstyle\setbox\@currbox\float@makebox\columnwidth\fi
  \box\@currbox\vskip\intextsep\relax}
```

`\float@dblend` The `\float@dblend` command finishes up double-column floats. This uses the same approach as `\float@end` above. It seems to work.

```
\newcommand\float@dblend{\@endfloatbox
  \global\setbox\@currbox\float@makebox\textwidth
  \let\@endfloatbox\relax\end@dblfloat}
```

4.3 Captions and lists of floats

Now for the caption routines. We use a box, `\@floatcapt`, to hold the caption while the float is assembled.

```
\newsavebox\@floatcapt
```

`\caption` This is the only LaTeX macro that this document style supersedes. Our `\caption` command checks whether there is a custom style defined for the current float (whose type can be found in `\@captype`). If so, the caption routines from float are used, otherwise (or when the float style has been applied via `\restylefloat*` so that captions aren't handled by this package) we call the vanilla LaTeX routines.

```
\renewcommand\caption{%
  \ifx\@captype\@undefined
    \@latex@error{\noexpand\caption outside float}\@ehd
    \expandafter\@gobble
  \else
    \refstepcounter\@captype
    \let\@tempf\@caption
    \expandafter\ifx\csname @float@c@\@captype\endcsname\relax\else
      \expandafter\expandafter\let
        \expandafter\@tempf\csname @float@c@\@captype\endcsname\fi\fi
  \@dblarg{\@tempf\@captype}}
```

`\float@caption` The `\float@caption` macro takes care of entering the caption into the appropriate listing. It also controls the typesetting of the caption itself, although a style-dependent macro `\@fs@capt` is called to handle the specifics. Note that because the caption is saved in a box instead of being output to the float right away, you cannot simply put a legend after the caption proper; it has to follow the `\caption` command in an optional argument.

First of all, we call `\addcontentsline` to update the list of floats of this class. Note that `\float@caption` is `\long` to allow for paragraph breaks in its arguments.

```
\long\def\float@caption#1[#2]#3{\addcontentsline{\@nameuse{ext@#1}}{#1}%
  {\protect\numberline{\@nameuse{the#1}}{\ignorespaces #2}}
```

Now we collect the caption proper. The caption name and number are taken from `\fnum@`⟨*float*⟩, where ⟨*float*⟩ is the class of float we're currently handling.

```
  \global\setbox\@floatcapt\vbox\bgroup\@parboxrestore
    \normalsize\@fs@capt{\@nameuse{fnum@#1}}{\ignorespaces #3}%
```

Finally we check for the presence of the optional argument. If there is one, we call `\float@ccon` to pick it up; otherwise, the `\egroup` finishes off the box.

```
    \@ifnextchar[{\float@ccon}{\egroup}}
```

`\float@ccon` The `\float@ccon` macro expands to the optional argument of a `\caption` command, followed by `\par\egroup`. Note that this precludes using `\verb` & Co. in the optional argument; the interested reader is urged to fix this problem as an exercise.

```
\long\def\float@ccon[#1]{#1\par\egroup}
```

`\listof` The `\listof` command reads the desired list of floats from the appropriate auxiliary file. The file is then restarted. First of all, we check whether the float style that's supposed to be listed is actually defined. If not, we output a `\float@error`.

```
\newcommand*{\listof}[2]{%
  \@ifundefined{ext@#1}{\float@error{#1}}{%
```

All's well until now. We define the `\l@`⟨*float*⟩ command that LaTeX needs for formatting the list, and then typeset the appropriate list header.

```
    \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
    \float@listhead{#2}%
```

Next we call `\@starttoc` with the correct file extension to do the actual work. If `\parskip` is non-zero, vertical space would be added between the individual list entries. To avoid this, we zero `\parskip` locally. This should be done after the `\float@listhead` above since `\parskip` also influences the spacing of headings, and the listings would look different from other chapters otherwise. (Suggested by Markus Kohm.)

```
    \begingroup\setlength{\parskip}{\z@}%
      \@starttoc{\@nameuse{ext@#1}}%
    \endgroup}}
```

`\float@listhead` This command generates the beginning of a list of floats. Currently the list appears at the chapter or the section level, depending on whether chapters are supported in the document class. According to a suggestion from Markus Kohm, this is now in a separate command so it can be overridden by other packages. We also use `\MakeUppercase` instead of `\uppercase`; when this piece of code was first written `\MakeUppercase` hadn't been invented yet, and for some reason this never got updated.

```
\providecommand*{\float@listhead}[1]{%
  \@ifundefined{chapter}{\def\@tempa{\section*}}%
    {\def\@tempa{\chapter*}}%
  \@tempa{#1\@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}}}%
```

`\float@addtolists` This command allows LaTeX programmers to add something to all currently-defined lists of floats, such as some extra vertical space at the beginning of a new chapter in the main text (`\float@addtolists{\protect\addvspace{10pt}}`), without knowing exactly which lists of floats are currently being constructed. This command currently does *not* operate on the `lot` and `lof` lists.

```
\newcommand\float@addtolists[1]{%
  \def\float@do##1{\addtocontents{##1}{#1}} \the\float@exts}
```

5 The Float Styles

Finally, we define the standard float styles that were outlined in the Introduction. Every float style ⟨*style*⟩ corresponds to a command `\fs@`⟨*style*⟩ which contains the definitions of the style commands, namely

`\@fs@pre`	top of the float
`\@fs@mid`	between float and caption
`\@fs@post`	bottom of the float
`\@fs@capt`	formatting routine for caption
`\@fs@cfont`	font for caption name & number

\floatc@plain The \floatc@plain macro formats a caption the way LaTeX does it: if the caption is fairly short, it is centered, otherwise it is formatted as a paragraph. The only difference is that the portion containing the caption name and number uses the \@fs@captfont.

```
\newcommand\floatc@plain[2]{\setbox\@tempboxa\hbox{{\@fs@cfont #1:} #2}%
  \ifdim\wd\@tempboxa>\hsize {\@fs@cfont #1:} #2\par
    \else\hbox to\hsize{\hfil\box\@tempboxa\hfil}\fi}
```

\fs@plain The plain float style is similar to what LaTeX does of its own accord; the only difference is that the caption is guaranteed to come out at the bottom of the float.

```
\newcommand\fs@plain{\def\@fs@cfont{\rmfamily}\let\@fs@capt\floatc@plain
  \def\@fs@pre{}\def\@fs@post{}%
  \def\@fs@mid{\vspace\abovecaptionskip\relax}%
  \let\@fs@iftopcapt\iffalse}
```

\fs@plaintop The plaintop float style is similar to the plain float style except that the caption comes out at the top.

```
\newcommand\fs@plaintop{\fs@plain
  \def\@fs@mid{\vspace\belowcaptionskip\relax}%
  \let\@fs@iftopcapt\iftrue}
\let\floatc@plaintop=\floatc@plain
```

\floatc@ruled The \floatc@ruled command is even simpler than the \floatc@plain macro. The caption is simply printed 'as is'.

```
\newcommand\floatc@ruled[2]{{\@fs@cfont #1} #2\par}
```

\fs@ruled In the ruled float style, the caption appears at the top of the float, preceded and followed by horizontal rules. Another rule followes the whole of the float.

```
\newcommand\fs@ruled{\def\@fs@cfont{\bfseries}\let\@fs@capt\floatc@ruled
  \def\@fs@pre{\hrule height.8pt depth0pt \kern2pt}%
  \def\@fs@post{\kern2pt\hrule\relax}%
  \def\@fs@mid{\kern2pt\hrule\kern2pt}%
  \let\@fs@iftopcapt\iftrue}
```

\fs@boxed The boxed float style puts the float into a box (which is slightly larger than the usual \textwidth). The caption appears below the box.

```
\newcommand\fs@boxed{\def\@fs@cfont{\bfseries}\let\@fs@capt\floatc@plain
  \def\@fs@pre{\setbox\@currbox\vbox{\hbadness10000
    \moveleft3.4pt\vbox{\advance\hsize by6.8pt
      \hrule \hbox to\hsize{\vrule\kern3pt
        \vbox{\kern3pt\box\@currbox\kern3pt}\kern3pt\vrule}\hrule}}}%
  \def\@fs@mid{\kern2pt}%
  \def\@fs@post{}\let\@fs@iftopcapt\iffalse}
```

Before we finish, we set the float style to plain.

```
\floatstyle{plain}
```

Other float styles can be added without much ado. If there are many more float styles, it should be worthwhile to invent a scheme similar to that used in Frank Mittelbach's theorem option in order to conserve space, i.e., put the float styles into individual files that can be loaded on demand. I would like to hear from people who define interesting float styles.

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.