Update: Published article exists

As requested by many people, this answers has been turned into a TUGboat article that was published a while ago (TUB 35/3, 2014). It can be read and downloaded as pdf from https://www.latex-project.org/publications (as well as a bunch of other papers and conf talks from the LaTeX Project Team).

===

To answer this question one has to first understand the basic rules that govern LaTeX's standard placement of floats. Once these are understood, adjustments can be made, for example, by modifying float parameters, or by adding certain packages that modify or extend the basic functionality.

LaTeX floats terminology

Float classes

Each float in LaTeX belongs to a class. By default, LaTeX knows about two classes, *viz.*, `figure` and `table` . Further classes can be added by a document class or by packages. The class a float belongs to influences certain aspect of the float's position, such as its default placement specification (if not overridden by the float itself).

One important aspect of the float placement algorithm is that LaTeX never violates the order of placement *within* a class of floats. E.g., if you have figure1, table1, figure2 in a document, then figure1 will always be placed before figure2. However, table1 (belonging to a different float class) will be placed independently and hence can appear before, after, or between the figures.

Float areas

LaTeX knows about 2 float areas within a column where it can place floats: the top area and the bottom area of the column. In two-column layout, it also knows about a top area spanning the two columns. There is no bottom area for page-wide floats in two-column mode

In addition, LaTeX can make float columns and float pages, i.e, columns and pages which contain only floats.

Finally LaTeX can place floats inline into the text (if so directed).

Float placement specifiers

To direct a float to be placed into one of these areas, a float placement specifier has to be provided as an optional argument to the float. If no such optional argument is given then a default placement specifier is used (which depends on the float class as mentioned above). A float placement specifier can consist of the following characters *in any order*:

- `!` indicates that some restrictions should be ignored (discussed later)
- `h` indicates that the float is allowed to be placed inline
- `t` indicates that the float is allowed to go into a top area
- `b` indicates that the float is allowed to go into a bottom area
- `p` indicates the the float is allowed to go on a float page or column area

The order in which these characters are stated does **not** influence how the algorithm tries to place the float (*e.g.,* [ht] or [th] will make no difference)! This is one of the common misunderstandings, for instance when people think that `bt` means that the bottom area should be tried first.

However, if a letter is not present then the corresponding area will not be tried at all.

Float algorithm parameters

There are about 20 parameters that influence the placement. Basically they define

- how many floats can go into a certain area,
- how big an area can become,
- how much text there has to be on a page (in other words, how much the top and bottom areas can occupy) and
- how much space will be inserted
  - between consecutive floats in an area and
  - between the area and the text above or below.

Float reference point

The point where the float is placed in the source document affects the placement of the float in the output because it determines when LaTeX sees the float for the first time. It's important to understand that if a float is placed in the middle of a paragraph, this reference point is the next line break, or page break, in the paragraph that follows the actual placement in the source.

Basic behavioral rules of LaTeX's float mechanism

With this knowledge, we are now ready to delve into the algorithm's behavior.

First we have to understand that all of LaTeX's typesetting algorithms are designed to avoid any sort of backtracking. This means that LaTeX reads through the document source, formats what it finds and (more or less) immediately typesets it. The reasons for this design choice were to limit complexity (which is still quite high) and also to maintain reasonable speed (remember that this is from the early eighties).

For floats, this means that the algorithm is *greedy*, i.e., the moment it encounters a float it will immediately try to place it and, if it succeeds, it will never change its decision. This means that it may choose a solution that could be deemed to be inferior in the light of data received later on.

For example, if a figure is allowed to go to the top or bottom area, LaTeX may decide to place this figure in the top area. If this figure is followed by two tables which are only allowed to go to the top, these tables may not fit anymore. A solution that could have worked in this case (but wasn't tried) would have been to place the figure in the bottom are and the two tables in the top area.

The basic sequence

So here is the basic sequence the algorithm runs through:

- If a float is encountered, LaTeX attempts to place it immediately according to its rules (detailed later)
  - if this succeeds, the float is placed and that decision is never changed;
  - if this does not succeed, then LaTeX places the float into a holding queue to be reconsidered when the next page is started (but not earlier).
- Once a page has finished, LaTeX examines this holding queue and tries to empty it as best as possible. For this it will first try to generate as many float pages as possible (in the hope of getting floats off the queue). Once this possibility is exhausted, it will next try to place the remaining floats into top and bottom areas. It looks at all the remaining floats and either places them or defers them to a later page (i.e., re-adding them to the holding queue once more).
- After that, it starts processing document material for this page. In the process, it may encounter further floats.
- If the end of the document has been reached or if a `\clearpage` is encountered, LaTeX starts a new page, relaxes all restrictive float conditions, and outputs all floats in the holding queue by placing them on float page(s).

Detailed placement rules when encountering a float

Whenever LaTeX encounters a float environment in the source, it will first look at the holding queue to check if there is already a float of the same class in the queue. If that happens to be the case, no placement is allowed and the float immediately goes into the holding queue.

If not, LaTeX looks at the float placement specifier for this float, either the explicit one in the optional argument or the default one from the float class. The default per float class is set in the document class file (e.g., `article.cls`) and very often resolves to `tbp`, but this is not guaranteed.

- If the specifier contains a `!`, the algorithm will ignore any restrictions related either to the number of floats that can be put into an area or the max size an area can occupy. Otherwise the restrictions defined by the parameters apply.
- As a next step it will check if `h` has been specified.
  - If so, it will try to place the float right where it was encountered. If this works, i.e., if there is enough space, then it will be placed and processing of that floats ends.
  - If not, it will look next for `t` and if that has been specified it will try to place the float in the top area. If there is no restriction that prevents it then the float is placed and processing stops.
  - If not it will finally check if `b` is present and, if so, it will try to place the float into the bottom area (again obeying any restrictions that apply if `!` wasn't given).
- If that doesn't work either or is not permitted because the specifier wasn't given the float is added to the holding queue.

- A `p` specifier (if present) is not used during the above process. It will only be looked at when the holding queue is being emptied at the next page boundary.

This ends the processing when encountering a float in the document.

Emptying the holding queue at the page boundary

After a page has been finished, LaTeX looks at the holding queue and attempts to empty it out as best as possible. For this it will first try to build float pages.

Any floats participating in a float page (or column) must have a `p` as a float specifier in its float placement specification. If not, the float cannot go on a float page and, on top of that, the absence of the `p` specifier will prevent any following floats of the same class to go there too!

If the float can go there, it will be marked for inclusion on the float page, but the processor may still abort the attempt if the float page will not get filled "enough" (depending on the parameter settings for float pages). Only at the very end of the document, or when a `\clearpage` has been issued, are these restrictions lifted, and a float will then be placed on a float page even if it has no `p` and would be the only float on that page.

Creation of float pages continues until the algorithm has no further floats to place or if it fails to produce a float page. In the latter case, all floats that have not been placed so far are then considered for inclusion in the top and bottom areas of the next page (or column).

The process there is the same as the one described above, except that

- the `h` specifier has no longer any meaning (as we are, by now, far away from the original "here") and is therefore ignored
- and the floats are this time coming not from the source document but one after the other from the holding queue.

Any float that couldn't been placed then is put back to the holding queue, so that when LaTeX is ready to look at further textual input from the document the holding queue may already contain floats. A consequence of this is that a float encountered in the document may immediately get deferred just because an earlier float of the same float class is already on hold.

Details on the parameters that restrict/influence the placement

There are four counters that control how many floats can go into areas:

- `totalnumber` (default 3) is the maximum number of floats on a text (!) page
- `topnumber` (default 2) is the maximum number of floats in the top area
- `bottomnumber` (default 1) is the maximum number of floats in the bottom area
- `dbltopnumber` (default 2) is the maximum number of full sized floats in two-column mode going above the text columns.

The size of the areas are controlled through parameters (changed with `\renewcommand`) that define the maximum (or minimum) size of the area, expressed as a fraction of the page height:

- `\topfraction` (default 0.7) maximum size of the top area
- `\bottomfraction` (default 0.3) maximum size of the bottom area
- `\dbltopfraction` (default 0.7) maximum size of the top area for double-column floats
- `\textfraction` (default 0.2) *minimum* size of the text area, i.e., the area that must *not*be occupied by floats

The space that separates floats within an area, as well as between float areas and text areas, is defined through the following parameters (all of which are rubber lengths, i.e., can contain some stretch or shrink components). Their defaults depend on the document font size and change when class options like `11pt` or `12pt` are used. We only show the 10pt defaults:

- `\floatsep` (default 12pt plus 2pt minus 2pt) the separation between floats in top or bottom areas
- `\dblfloatsep` (default 12pt plus 2pt minus 2pt) the separation between double-column floats on two column pages
- `\textfloatsep` (default 20pt plus 2pt minus 4pt) the separation between top or bottom area and the text area
- `\dbltextfloatsep` (default 20pt plus 2pt minus 4pt) the analog of `\textfloatsep` for two-column floats

For inline floats (that have been placed "here") the separation to the surrounding text is controlled by

- `\intextsep` (default 12pt plus 2pt minus 2pt)

In case of float pages or float columns (i.e., a page or a column of a page containing only floats) parameters like `\topfraction` etc. do not apply. Instead the creation of them is controlled through

- `\floatpagefraction` (default 0.5) minimum part of the page (or column) that need to be occupied by floats to be allowed to form a float page (or column)

Consequences of the algorithm

A float may appear in the the document earlier than its placement in the source

The placement of the float environment in the source determines the earliest point where it can appear in the final document. It may move visually backward to some degree as it may be placed in the top area on the current page. It can, however, not end up on an earlier page as the surrounding text due to the fact that LaTeX does no backtracking an the earlier pages have already been typeset.

Thus normally a float is placed in the source near its first call-out (i.e., text like "see figure 5") because this will ensure that the float appears either on the same page as this text or on a later page. However, in some situations you may want to place a float on the preceding page (if that page is still visible from the call-out). This is only possible by moving the float in the source.

Double-column floats are always deferred first

When LaTeX encounters a page-wide float environment (indicated by a `*` at the end of the environment name, e.g., `figure*`) in two column-mode, it immediately moves it to the deferred queue. The reason for this behavior again lies in the "greedy" behavior of its algorithm: if LaTeX is currently assembling the second column of that page, the first column has already been assembled and stored away; recall that because LaTeX does not back-track there is no way to fit the float on the current page. To keep the algorithm simple, it does the same even if working on the first column (where it could in theory do better even without back-tracking).

Thus, in order to place such a float onto the current page, one has to manually move it to an earlier place in the source -- before the start of the current page. If this is done, obviously any further change in the document could make this adjustment obsolete; hence, such adjustments are best done (if at all) only at the very last stage of document production --- when all material has been written and the focus is on fine-tuning the visual appearance.

There is no bottom float area for double-column floats

This isn't so much a consequence of the algorithm but rather a fact about it. For double-column floats the only possible placements are the top area or a float page. Thus if somebody adds an `h` or a `b` float placement specifier to such a float it simply gets ignored. So `{figure*}[b]` implies that this float will not get typeset until either a `\clearpage` is encountered or the end of the document is reached.

All float parameters (normally) restrict the placement possibilities

This may be obvious but it is worth repeating: any float parameter defines a restriction on LaTeX's ability to place the floats. There is always a way to set a parameter in such a way that it does not affect the placement at all. Unfortunately, in doing so one invites rather poorly looking placements.

By default LaTeX has settings that are fairly liberal. For example, for a float page to be accepted the float(s) must occupy at least half of the available page. Expressed differently, this means that such a page is allowed to be half empty (which is certainly not the best possible placement in most cases).

What often happens is that users try to improve such settings and then get surprised when suddenly all floats pile up at the end of the document. To stay with this example: if one changes the parameter `\floatpagefraction` to require, say, `0.8` of the float page, a float that occupies about `0.75` of the page will not be allowed to form a float page on its own. Thus, if there isn't another float that could be added and actually fits in the remaining space, the float will get deferred and with it all other floats of the same class. But, even worse, this specific float is too big to go into the next top area as well because there the default maximum permissible area is `0.7`. As a result all your floats stay deferred until the next `\clearpage`.

For this reason it is best not to meddle with the parameters while writing a document or at least not to do so in a way that makes it more difficult for the algorithm to place a float close to its call-out. For proof-reading it is far more important to have a figure next to the place it is referenced then to avoid half-empty pages. Possibilities for fine-tuning an otherwise finished document are discussed below.

Another conclusion to draw here is that there are dependencies between some of the float parameters; it is important to take these dependencies into account when changing their values.

"Here" really just means "here if it fits"

... and often it doesn't fit. This is somewhat surprising for many people, but the way the algorithm has been designed the `h` specifier is not an unconditional command. If an unconditional command is needed, extension packages such the `float` package offer `H` as an alternative specifier that really means "here" (and starts a new page first if necessary).

Float specifiers do not define an order of preference

As mentioned above, the algorithm tries to place floats into available float areas in a well-defined order that is hard-wired into the algorithm: "here", "top", "bottom" and -- on page boundaries -- first "page" and only if that is no longer possible "top" followed by "bottom" for the next page.

Thus specifying `[bt]` does *not* mean try bottom first and only then top. It simply means allow this float to go into top or bottom area (but not onto a float page) just like `[tb]` would.

Relation of floats and footnotes

This is not exactly a consequence of the algorithm but one of its implementation: Whenever LaTeX tries to decide on a placement for a float (or a `\marginpar` !) it has to trigger the output routine to do this. And as part of this process any footnotes on the page are removed from their current place in the galley and are collected together in the `\footins` box. After placing the float (or deferring it) LaTeX then returns the page material to the galley, but because of the Output Routine behavior the galley has now changed: LaTeX has to put the footnotes somewhere but all in one place. What it does is it reinserts the footnotes (the `\footnotetext` to be precise) at the end of the galley. There are some good reasons for doing this, one of which is that LaTeX expects that all of the returned material still fits on the current page.

However, if for some reason a page is finally taken at an earlier point then the footnotes will show up on the wrong page or column. This is a fairly unlikely scenario but if it happens check if there is a float near the chosen page break and either move the float or guide the algorithm by using explicit page breaks. and example can be found in this question.

In fact that particular case is worth highlighting: Do **not** place a float directly after a heading, unless it is a heading that always starts a page. The reason is that headings normally form very large objects (as they prevent page breaks directly after it). However placing a float in the middle of this means that the Output Routine gets triggered before LaTeX makes its decision where to break and any footnotes get moved into the wrong place

Documentation of the algorithm

As requested here is some information on existing documentation. The algorithm and its implementation is documented in the file `ltoutput.dtx` as part of the LaTeX kernel source. This can be typeset standalone or as part of the whole kernel (i.e., by typesetting `source2e.tex` --- ignore the checksum error, sorry).

This documentation is an interesting historical artifact. Parts of it show semi-formatted pseudo code which dates back to LaTeX2.09; in other words it is from the original documentation by Leslie Lamport. The actual code is documented using doc style and in parts is more or less properly documented (from scratch) and dates back to 1994 or thereabout when Chris Rowley and myself adjusted and extended the original algorithm for LaTeX2e. It also fairly openly documents the various issues with the algorithm and/or its implementation --- in many cases we didn't dare to alter it because of the many dependencies and, of course, because of the danger to screw up too many existing documents that implicitly rely on the current behavior for good or worse. Near the end you'll find a list of comments compiled back then on the algorithm, but there are also comments, questions, and tasks (? :-) sprinkled throughout the documentation of the code.

One interesting aspect of this file (that I forgot all about) is that it contains all code necessary to trace the behavior of the algorithm in real life. Unfortunately, I never made this officially available or so it seems. It would probably need a good amount of cleanup and better formatting of the tracing output it produces to be usable for the general public. But even in its current form it does give some interesting insight in the behavior of the algorithm and how certain decisions come about.

So if somebody wants to play with it or wants to trace some strange float placements, then all that is necessary to do (fingers crossed) is to make a short file `fltrace.ins` with the following content:

```
\input docstrip

\generateFile{fltrace.sty}{t}{%
  \from{ltoutput.dtx}{fltrace,trace}
}

\endbatchfile
```

Run through LaTeX this will produce the style file `fltrace.sty`. You can then use this in your documents via

```
\usepackage{fltrace}
\tracefloats               % start float tracing
```

The command `\tracefloatvals` displays the current state of several float parameters and with `\notrace` tracing is turned off again. As already mentioned this is not an official package, but it might be useful in one or the other situation or purely out of interest after having studied the documentation.

How to address specific issues

to be added later (and/or perhaps reference to other questions)

Info For the moment I guess I'm done. The last section will need material but during the next days I'm certainly not able to provide it -- so if anybody feels like fixing my English ... please go ahead. It might be also a good approach to collect here references to other questions addressing specific topics. **Info End**

share improve this answer

edited Apr 13 at 12:36
Community ◆
1

answered Dec 21 '11 at 13:58
Frank Mittelbach
55.2k 4 161 238

1 @Frank I just want to make a suggestion. It should be nice if you c
control the actual positioning of the floats. For example, if there a
queue that would fit side-by-side on the current page, how can o
by-side. Please note that I don't mean how to do this using a spec
using the standard float environments without user-intervention (
extra lines of code to provide the rules). – user10274 Dec 28 '11 a

1 @Marc unfortunately the answer to that is simple: there is no way
believe me a smarter algorithm is far from simple to provide. For L
OR that does fairly general float positioning together with balanci
to be used --- one of Jospeh's tasks to redo and make better ;-) –
15:57

1 @FrankMittelbach What about the `flafter` package? – Jubobs Oc

@Jubobs that would belong in the last section (which I still have to
Frank Mittelbach Oct 31 '13 at 21:08

@FrankMittelbach Sorry for the late comment. This is as good a wr
subject. However, taking the risk of sounding ignorant, what does
1 '14 at 7:40

@VaibhavGarg don't think this is ignorance, more me assuming to
Routine – Frank Mittelbach Mar 3 '14 at 20:37

Is there some strong, specific typesetting reason why double-colu
bottom of the page? If not, is there some workaround to permit th

@episanty no strong reason, it is simply not implemented and bac
memory considerations. Package stfloats adds this, perhaps I shou
Frank Mittelbach Jul 15 '14 at 16:38

Thanks, I'll give it a look. That package may solve the problem in t
16:52

In the LNI template, I found `\def\fps@figure{htbp}` which seem
for float parameters. Think, this is worth including in the answer, i
8:09