The `latex-lab-footnotes` code*

Frank Mittelbach, LaTeX Project

January 25, 2026

Abstract

to be written

Contents

*

1 Introduction

This code reimplements the footnote interfaces for LaTeX offering configurable methods for layout and functionality adjustments that avoid overwriting each other when used in classes as well as in packages (as far as possible — obviously some adjustments are mutually exclusive). This is achieved by providing a larger number of hooks (for areas where different packages/classes can easily coexist with their adjustments) and a number of sockets to which only one class or package can write to successfully (in case of multiple changes the last one wins). The latter are for special functionality, e.g., if footnote text is typeset as a single paragraph, it can't be configured the same time to be typeset vertically with one footnote below each other.

The interfaces are set up to support tagged PDF, but in order for this to work, all packages altering the footnote setup should use the interfaces provided here and not do it through the legacy methods (though there is some support for the latter as well, but it will not work in all cases).

1.1 Configuration methods

Historically, the footnote setup in LaTeX was done by providing definitions for `\@makefnmark` (format the footnote mark in running text and in front of the footnote text) and `\@makefntext` (formatting the footnote text and placing a mark in front of it).

There was a default definition for `\@makefnmark` in the format that was used by most document classes, but `\@makefntext` had to be defined in the class itself because the format didn't provide a default. As a result you will find definitions for the latter in all document classes and definitions for `\@makefnmark` only in very few.

Furthermore, to enable special footnote layouts or provide additional functionality a few packages (and a few classes) overwrote other internal commands of LaTeX's footnote mechanism. The commands affected in this way are mainly `\@footnotemark` and `\@footnotetext`. These overwrites could not be used in combination, so either the packages/classes had to be aware of being loaded together (which they sometimes did or tried to) or they would fail by overwriting each other unconditionally.

The present rewrite is an attempt to improve this situation, but of course, it will only work if all packages/classes make use of the new interfaces. Fortunately, the number of problematical packages altering these internal commands are fairly small so arranging for updates is a realistic goal — to achieve properly tagged PDF it is a requirement.

2 Sockets and hooks

We use sockets for those parts that can be controlled only by one package or by the kernel and hooks for places where it may be possible that several packages or the document class adds code (typically declarations such as font changes, etc.).

Note that sockets are of interest only to very few specialized packages, mainly footmisc, and packages providing similar functionality—the current documentation is therefore fairly sketchy.

In contrast the hooks are of interest to many classes to provide their layout alterations in a way that it works smoothly with other packages handling aspects of footnote formatting.

2.1 Formatting the mark in the main text

This implements formatting the mark[1] and its relation to surrounding text, e.g., if several marks appear in the same place, etc.

2.1.1 Sockets

None: everything is implemented through a single definition for `\@footnotemark` that offers a number of hooks that can be used by packages to implement handling of multiple marks and the formatting of marks.

2.1.2 Hooks for formatting the footnote mark in text

The hooks to customize the marks in the text are the following:

[1] Like this one.

`fnmark/before`

Executed at the very beginning of `\footnotemark`. Currently there are two packages (bibarts and chextras) that prepend material at this point (not necessarily correctly, e.g., they do not all check that they are in horizontal mode).

This hook is paired with hook `fnmark/after`.

`fnmark`

Executed in horizontal mode and after the current space factor has been saved away for reuse. This is where currently code for multiple marks does its preparation (as done by footmisc and others).

The hook is only executed in hmode, i.e., not if the mark is generated in math — maybe that means the multiple handling should happen later?

After the hook a `\nobreak` is executed, so any "material" added in the hook is tied to the following mark unless it contains its own permissible penalty.

`fnmark/begin`

This hook is executed directly in front of the typeset mark. This is the place where hyperref would have added part of its code, i.e., after the `\nobreak` mentioned above. With the integration of hyperlinks in the tagging code this hook may not be necessary at all.

`fnmark/end`

This hook is executed directly after the typeset mark. It is used by memhfixc, scrlttr2, and footmisc. Used, for example, to implement support for multiple marks in succession.

It is *not* a reversed hook.

`fnmark/after`

This hook is executed at the very end of the `\footnotemark` command.

It is a reversed hook to pair with `fnmark/before`

2.1.3 Additional configuration possibilities

The actual formatting is done through `\@makefnmark` — no special customization support for now.

2.2 Formatting the footnote text

This implements the formatting of the footnote text the way it appears at the bottom of the page (default case), or possibly elsewhere, e.g. in the margin.

2.2.1 Sockets

To cater for different layout configurations there are four sockets that can be set by a package or class but there should be only one per document setting them, i.e., if two packages/classes set them they are mutually incompatible (or rather the last one wins most likely). These are:

`fntext/process` (1 argument)

This socket receives all material that is to be processed (or stored) including color protection code and what have you. The `default` executes `\insert\footins`.

Available plugs are `default`, `side` (side notes), and `mp` (minipage).

`fntext/make` (1 argument)

This socket receives the ⟨*text*⟩ as given in `\footnote` or `\footnotetext` in the document and adds formatting instructions to it.

The `default` plug runs `\@makefntext` which contains various hooks for customization. For most scenarios this is sufficient. However, when running all footnotes as a single paragraph at the bottom, then each footnote needs to be prepared prior to storing it with `\insert` and this socket allows running extra code to do that.

Available plugs are `default` and `para`.

`fntext/begin` (no argument)

The socket is executed near the start of the argument for the `fntext/make` socket. By `default` it adds a strut to the footnote material so that consecutive footnotes are properly spaced vertically. In some use cases this is not appropriate (e.g., when running all footnotes as a single paragraph) and so with this socket one can cancel the action or do something else instead.

Available plugs are `default` and `noop`.

`fntext/end` (no argument)

This socket is executed at the very end of the argument passed to socket `fntext/make`. By `default` it adds a final strut as long as we are still in horizontal mode (i.e., processing the footnote text paragraph). When running several footnotes in one paragraph some additional material (some horizontal glue) needs adding at this point which is done with the plug `para`.

Available plugs are `default`, `para`, and `noop`.

All standard plugs for the socket `fntext/make` run `\@makefntext` and this command contains two further sockets (unless it is overwritten by a legacy class):

`fntext/mark` (0 arguments)

This socket has no input arguments but uses `\@makefnmark` to typeset the mark in front of the footnote text. Its `default` uses code that examines the value of `\footnotemargin` and based on its setting typeset the mark in different ways:

- positive: typeset the mark in a box of that size
- zero: use `\llap` around the mark
- negative: use `\llap` but with a box of the given size negated inside
- `-\maxdimen`: just use `\@makefnmark`

For most cases this would be flexible enough, but if not then a class can define its own plug to specify the placement of the mark.

Available plugs are `default` and `noop` (no mark is produced).

fntext/text (1 argument)

This socket manages the formatting of the footnote text (presented as an argument) once the mark has been typeset. In all cases we can think of this formatting is better configured via the available hooks described below, so the `default` just grabs the argument and processes it without any other action. It is really only there to allow for some fancy stuff that some design comes up with.

Available plugs are `identity` (default) and `noop`.

The above configuration points are sufficient to implement all commonly used footnote layouts assuming L-R typesetting. For R-L typesetting they or may or may not need some extension (though that is not clear right now).

2.2.2 Hooks for formatting the footnote text

fntext/before

Executed at the very beginning of `\footnotetext`. Currently there is one package (linguex) that prepends material at this point.

This hook is paired with hook `fnmark/after`.

fntext

Executed at the beginning of the material passed to the first configuration point. Typically used to set any baseline stretch for the footnote text, e.g., by setspace, footmisc, `uathesis.cls` and others. Could be done in a later hook but is a bit more efficient here.

After the hook has run, the font is established, i.e., it can't be used to set a different font size.

fntext/para

After the font is set (after the previous hook), some default paragraph parameters are set up including `\interlinepenalty`, `\hsize`, `\parindent` and a number of others, as some of them depend on the font size. Then the `fntext/para` is run which can overwrite the default. If one wants to change the font size, it is probably necessary to reset these other parameters too, e.g., `\parindent`, which can be done here.

Note: the socket `fntext/make` normally runs the command `\@makefntext` or some code that eventually runs this command, and this then produces the footnote mark in front of the formatted footnote text. In front of both the mark and the footnote text some classes have placed paragraph parameter adjustments in their redefinition of `\@makefntext`. However, there is no need to place it there it could equally well go into the `fntext/para` hook. We therefore do not provide another hook at this other point.

fntext/begin & fntext/end

The footnote text itself is surrounded by the hooks `fntext/begin` and `fntext/end`. The two hooks are not paired as they are typically used independently.

fntext/after

At the very end of `\footnotetext` we execute the hook `fntext/after` which is a reversed hook paired with `fntext/before`. Some packages, e.g., linguex, have code in that position.

2.2.3 Additional configuration possibilities

The formatting of the footnote mark in front of the footnote text is influenced by the setting of the dimen parameter `\footnotemargin`. By default its value is 1.8em in the current text font (or `-\maxdimen` when the para option is chosen). The following rules apply:

- If it has the value `-\maxdimen` then the mark is generated by `\@makefnmark`.
- Otherwise, if the value is negative then the mark is placed into an `\llap` left aligned in a box of size `-\footnotemargin`.
- If the value is zero an `\llap` is used without an inner box.
- If the value is greater zero (but less than `\maxdimen`) the mark is placed right aligned into a box of size `\footnotemargin`.
- The value `\maxdimen` is used as a marker to indicate that no value was given and that the default should be used, i.e. 1.8em or `-\maxdimen` depending on the chosen option.

2.3 Debugging sockets and hooks

For some rudimentary debugging we currently have `\DebugFNotesOn` (and `\DebugFNotesOff`). At the moment `\DebugFNotesOn` only shows the current settings for hooks and sockets related to the footnote code and then automatically turns itself off again.

3 Tagging and hyperlinking support

TODO: *this section needs work (and probably csname changes)*

Footnotes consist of a *footnotemark* (short: mark) that is typically placed in the text as a superscript number like this[1], and a *footnotetext* (short: note) that is placed at the bottom of the page. The *footnotetext* normally repeats at the begin the mark as a visual clue.

Tagging (and hyperlinking) has to connect the mark with the note. For the tagging code, we assume that every mark has exactly one associated note, and that every note is associated to at least one mark and can have more associated marks.

The mark doesn't need to be visible, e.g. the typesetted mark[1–3] denotes three marks, where the second is invisible. Tagging should produce here probably three `Lbl` structures (one without content), and an artifact for the range marker. If such a range is used, links can only point to the notes 1 and 3 and one has to suppress the linking for the second mark. This means that links and tagging are also related to the actual formatting of the footnote mark. In the following this problem is mostly ignored for now, but should not be forgotten and handled later.

3.1 Technical details for the tagging

The following sockets are set up for kernel use, when doing tagging: There name and/or function will probably change as they currently mix tagging with the link support.

TODO: review this sockets

tagsupport/fnmark (2 argument)

The socket is used in `\@footnotemark`/`\fnote_footnotemark:` and takes `\@makefnmark` or — if link support is wanted — `\socket_use:n{fnmark/link}{\@makefnmark}` as second argument. It prints the mark in the text and surrounds it with a tagging structure.

TODO: *describe and decide on names*

tagsupport/fntext/begin (no argument)

This socket is used before the main processing socket (so before the `\insert` command). It opens the FEnote structure.

tagsupport/fntext/end (no argument)

This socket is used after the main processing socket (so after the `\insert` command). It closes the FEnote structure.

tagsupport/fntext/mark (2 argument)

This socket is used around the mark in the footnote text. It adds tagging support.

tagsupport/fntext/text (2 argument)

This socket handles mc-chunks around the text of the footnote. It takes as second argument the text.

The *footnotemark* should create a `/Lbl` structure[2] that should contain a `/Ref` entry pointing to the structure of the *footnotetext*.

The *footnotetext* should create a `/FENote` [3] structure with a `/Ref` entry pointing to the structures of *all* marks related to the note. The mark at the begin of the note is in a `/Lbl`[4] structure but has to fulfil no special requirements.

Structure objects and the underlying properties used by the tagging code are initialized when the structure is opened. This means that one can not directly add data to a future structure but as structure objects are written at the end of the document it is possible to update `/Ref` entries in an end document hook.

So tagging has to solve two problems:

- the mark and the footnote text must be surrounded by the correct structure and marked content commands. This is not trivial as there are various layouts (bottom, marginpar, minipage) and the tagging from the automatic paratagging must be taken into account if one want to avoid faulty nesting.
- It must detect which marks are related to which notes so that it can setup the `/Ref` cross-references.

[2] to make it easier to identify the role we use `/footnotemark` which we rolemap to `/Lbl`
[3] We tag it as `/footnote` and role map it.
[4] We tag it as `/footnotelabel`.

3.2 Requirements for links

Links should go from the mark to the note. Sometimes it has been requested that links go back too, but as there can be more than one mark connected to a note it is not clear how to decide to which mark it should go. Using the keys from the PDF viewer to go back is normally better.

Links are closely related to the references stored in the `/Ref` entry of a mark and so are handled in the code together with them. But there are subtle technical differences to take care of as links and destinations are whatsits and so must be created at the correct time.

It should be possible to suppress the links both globally and locally.[5]

Footnote links should work also if tagging is disabled, either locally (e.g. in an artifact) or if tagging is deactivated globally. They should therefore use their own system to store the relations between mark and note, even if this duplicates some code.

3.3 The algorithmus to connect marks and notes

The connection is made by comparing the value of `\@thefnmark`.

The standard mark commands (`\footnotemark` and `\footnote`) store the current value of `\@thefnmark` with a combination of their own structure number (for the tagging) and a unique id (for the link) as a key in a property.

A following `\footnotetext` compares its own `\@thefnmark` with the values in the prop. If there is one or more match it stores the keys and removes the entries from the property (so in a normal document the property will never contain more than a few entries).

This works well as long as the `\footnotemark` commands are issued before the `\footnotetext` and as long as nothing unusual is done to `\@thefnmark`. It also works if a document uses more than one footnote series as long as they have distinct numbering systems, but in case a distinction is needed it is possible to define a new class with its own data structure and to switch locally to use this class. The following three commands are used for this.

The default class uses the name `default`

`\g_fnote_id_int` This integer is used as unique identifier to store data for a footnote. It should be increase when a footnote mark or a footnote text is created.

`\fnote_class_new:nn` `\fnote_class_new:nn{`⟨*name*⟩`}{`⟨*key/value option*⟩`}`

This declaration sets up the needed data structure. Currently this only consists of a property list which is used to store and manage the mark values. There are no options yet.

`\fnote_mark_struct_gput:nnn` `\fnote_mark_struct_gput:nnn{`⟨*mark*⟩`}{`⟨*class name*⟩`}`
`\fnote_mark_id_gput:nnn` `\fnote_mark_id_gput:nnn{`⟨*mark*⟩`}{`⟨*class name*⟩`}`

These commands store either the current structure number or the current id as key and the ⟨*mark*⟩ as value in the property list associated with the ⟨*class name*⟩.

[5] Currently hyperref only offers the option to suppress the footnote links globally with the option `hyperfootnotes=false`. To suppress them locally only the `NoHyper` environment is provided.

\fnote_mark_gpop_all:nnnN `\fnote_mark_gpop_all:nnN{⟨id or struct⟩}{⟨mark⟩}{⟨class name⟩}⟨sequence⟩`

This command stores all the keys/structure numbers whose value in the property list for ⟨*class name*⟩ are equal to ⟨*mark*⟩ into the sequence ⟨*sequence*⟩ and then removes them from the property list. The content of the sequence can then be used to create link targets and references.

3.3.1 `\footref`

`\footref` uses internally the same command to set the mark as `\footnotemark`, it only defines `\@thefnmark` differently. This `\@thefnmark` is not suitable for the method described above: as it contains a reference command it can't be used to match a note, also `\footref` can be used after the note has already been set. `\footref` disables therefore the automatic detection.

Instead the `\label` command is extended in the `\footnotetext` command to also store the structure number and `\footref` retrieves this number to setup the reference and the link.

The structure related to the `\footref` is added to the end of the `/Ref` array of the note and so the `/Ref` array doesn't necessarily reflect the order of the marks in the document. It would probably be possible to change this, but it is not clear if it actually matters and so it worth the additional coding and processing.

3.3.2 `\footnotemark` after `\footnotetext`

The automatic detection doesn't work if a `\footnotemark` is issued after the `\footnotetext` it refers to. There will be no error, but neither the link nor the `/Ref` will connect both.

The simple way to handle this is to use a label and `\footref`:

```
\footnotetext{\label{fn:a}text} ... \footref{fn:a}
```

An alternative would be to extend the syntax of `\footnotemark` and `\footnotetext` to allow to add a label which can then be used. For example

```
\footnotetext[label=fn:a]{text} ... \footnotemark[label=fn:a]
```

As both have already an optional argument, that requires the optional argument extension.

3.4 Links

The keys containing the structure number/id combination detected for the `/Ref` are also used for links. For links the second part is used as this id is increased also if tagging is deactivated.

A `\footnotetext` creates a bunch of destinations (in most cases this sums up to two destinations): one for every structure number in the `/Ref` (used as target by the mark commands) and one for the structure number of the footnotetext itself (used as target by `\footref`s commands).

3.5 Implementation details regarding tagging

3.6 Handling the mark

The mark in the text is handled by assigning an appropriate plug to the socket `tagsupport/fnmark`. It takes one argument, `\@makefnmark`, the command which formats the mark, and surrounds it by link and tagging commands. At the point where the socket is executed, `\@thefnmark` has already been defined and can be used to setup the reference detections.

3.7 Handling the footnotetext

The main part is done by assigning a different plug to socket `tagsupport/fntext/begin` and `tagsupport/fntext/end` surrounding the footnote text. These sockets are used to start and end the structure and attempt to detect to which mark the note is related.

The actual typesetting of the note text is done by `\fnote_makefntext:n` (or its LaTeX 2_ε name `\@makefntext`). In the new implementation this contains two further kernel sockets for tagging: `tagsupport/fntext/mark` and `tagsupport/fntext/text`. They get plugs assigned that add the tagging commands around note mark and note text.

3.8 Footnotes in minipages

In minipages the `\footnote` command uses a special marker (small italic letters by default) and puts the footnote text at the bottom of the box. The `\footnotemark` command uses the standard footnote counter and marker (and so typically creates a superscript number). It is meant to be used with a `\footnotetext` *outside* the minipage to create a footnote mark which refers to a footnote text at the bottom of the page. This means to repeat a footnote marker in a minipage you should use the `\footref` command.

Tagging works quite similar to normal footnotes if the new definition is used and if the minipage code is changed to use the new configuration point. The main problem here is currently the tagging of the minipage itself.

4 TODOs

- Special formatting of footnote marks in the text, e.g. if ranges or commas are used require special care as they should normally mark up such text as artifacts and perhaps have to insert empty structures to represent an invisible mark. This must be coordinated with the relevant packages and classes.
- manyfoot doesn't work correctly and must be analyzed.
- memoir is not supported at all and errors when the code tries to patch `\@makefntext`.

To be documented

5 The Implementation

All this is very rough and misses a lot of documentation.

```
1 ⟨*kernel⟩
2 ⟨@@=fnote⟩
```

5.1 File declaration

```
\ProvidesFile{latex-lab-footnotes.ltx}
             [\ltlabfootnotedate\space v\ltlabfootnoteversion\space
               changes to the footnote interfaces]
```

5.2 code not fully handled yet

```
%
% latex.ltx
% not looked at yet
% \@mpfootnotetext is probably no longer needed, or only to support other
% classes and package. See below about the minipage code.
%
% \long\def\@mpfootnotetext#1{%
%  \global\setbox\@mpfootins\vbox{%
%    \unvbox\@mpfootins
%    \reset@font\footnotesize
%    \hsize\columnwidth
%    \@parboxrestore
%    \def\@currentcounter{mpfootnote}%
%    \protected@edef\@currentlabel
%         {\csname p@mpfootnote\endcsname\@thefnmark}%
%    \color@begingroup
%      \@makefntext{%
%        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
%    \par
%    \color@endgroup}}
% ========
% used by the minipage footnote code.
%
% \def\@mpfn{footnote}
% \def\thempfn{\thefootnote}
% =========
% this perhaps need some configuration options.
%
%\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
%
% =========
%% alterations not covered:
%
% ./arabtex/afoot.sty  --- too different (and probably too old)
%
% =====
% alterations of footnotetext not covered:
%
% ./revtex4-1/revtex4-1.cls  ./revtex/ltxutil.sty ./revtex/revtex4-2.cls ... (need analysis)
% ./bigfoot/bigfoot.sty
%
% memoir needs checking too
%
% =====
%
% use of kerns to mark h-mode positions (unit sp)
```

```
%
% 1 = CJK
% 2 = CJK
% 3 = multiple footnotes (footmisc, koma, eledmac, tufte, memoir,
%    parnotes, sidenotes)
% 3 = outer kern in letter spacing (letterspace)
% 3 = beginning of list (examdesign.cls)
% 4 = CJK pigin
% 5 = CJK ruby

% 1-4 = polyglossia for korean
%
```

```
\ExplSyntaxOn
```

5.3 Temporary variables

```
\prop_new:N \l__fnote_tmpa_prop
\tl_new:N   \l__fnote_tmpa_tl
```

5.4 Public variables

A footnote mark will store its structure number (key) and the expanded `\@thefnmark` in this prop so that a following note can retrieve this info if needed. It is possible to use more than one footnote series (type) if needed (if different footnotes/note use the same numbering system). If this command is changed an accompanying property must be created

`\l_fnote_type_tl`

```
\tl_new:N  \l_fnote_type_tl
\tl_set:Nn \l_fnote_type_tl {default}
```

(*End of definition for* `\l_fnote_type_tl`.)

It must be possible to suppress the hyperlinking, both locally and globally. hyperref's hyperfootnotes option should set the boolean.

`\l_fnote_link_bool`

```
\bool_new:N       \l_fnote_link_bool
\bool_set_true:N  \l_fnote_link_bool
```

(*End of definition for* `\l_fnote_link_bool`.)

A hyperlink should have an changeable link type. This can be e.g. used to change the color or the border.

`\l_fnote_link_type_tl`

```
\tl_new:N  \l_fnote_link_type_tl
\tl_set:Nn \l_fnote_link_type_tl {link}
```

(*End of definition for* `\l_fnote_link_type_tl`.)

5.5 Internal variables

\l__fnote_linktarget_tl This command stores the name of a linktarget/destination when needed

```
\tl_new:N \l__fnote_linktarget_tl
```

(*End of definition for* `\l__fnote_linktarget_tl`.)

\l__fnote_currentlabel_tl This command is used to pass a label name around.

```
\tl_new:N \l__fnote_currentlabel_tl
```

(*End of definition for* `\l__fnote_currentlabel_tl`.)

\l__fnote_currentrefs_seq This sequence stores the list of reference of a note

```
\seq_new:N  \l__fnote_currentrefs_seq
```

(*End of definition for* `\l__fnote_currentrefs_seq`.)

The connection between the mark(s) in the text and the note is either deduced automatically or done through an label. The default is automatic, but we must be able to suppress it. For this we use a boolean.

\l__fnote_autodetect_bool

```
\bool_new:N       \l__fnote_autodetect_bool
\bool_set_true:N  \l__fnote_autodetect_bool
```

(*End of definition for* `\l__fnote_autodetect_bool`.)

This is used to pass the structure number of the note around, e.g. to a label inside the note.

```
\tl_new:N  \l__fnote_currentstruct_tl
\tl_set:Nn \l__fnote_currentstruct_tl {2}
```

5.6 Variants

```
\cs_generate_variant:Nn \hook_gput_code:nnn{nne}
\cs_generate_variant:Nn \tag_struct_use:n {e}
```

5.7 Updating `\@thefnmark`

\fnote_step_fnmark:nn This command updates `\@thefnmark`. The first argument is an optional integer expression, the second a counter name. If the optional argument is not given it steps the counter.

```
\cs_new_protected:Npn \fnote_step_fnmark:nn #1#2 {
  \tl_if_novalue:nTF {#1}
    {
      \stepcounter {#2}
      \protected@xdef \@thefnmark { \use:c { the#2 } }
    }
    {
     \group_begin:
```

Note that this is a local assignment even though LaTeX counters are normally globally changed. This is the way it was in 2e and so far we haven't changed it. The alternative would be to store the current value and restore it after `\@thefnmark` is altered.

```
        \int_set:cn { c@#2 }{ #1 }
        \unrestored@protected@xdef \@thefnmark { \use:c { the#2 } }
      \group_end:
    }
}
```

(*End of definition for* `\fnote_step_fnmark:nn`.)

`\fnote_set_fnmark:nn` This is similar to the previous command, but it doesn't step the counter but use the current value.

```
\cs_new_protected:Npn \fnote_set_fnmark:nn #1#2 {
  \tl_if_novalue:nTF {#1}
    {
      \protected@xdef \@thefnmark { \use:c { the#2 } }
    }
    {
     \group_begin:
        \int_set:cn { c@#2 }{ #1 }
        \unrestored@protected@xdef \@thefnmark { \use:c { the#2 } }
     \group_end:
    }
}
```

(*End of definition for* `\fnote_set_fnmark:nn`.)

5.8 Hooks

`fnmark/before` (*hook*)
`fnmark/after` (*hook*)
`fnmark` (*hook*)
`fnmark/begin` (*hook*)
`fnmark/end` (*hook*)

Hooks in the footnotemark command.

```
\NewMirroredHookPair{fnmark/before}{fnmark/after}
\NewHook{fnmark}
\NewHook{fnmark/begin}
\NewHook{fnmark/end}
```

`fntext/before` (*hook*)
`fntext/after` (*hook*)
`fntext` (*hook*)
`fntext/begin` (*hook*)
`fntext/end` (*hook*)
`fntext/para` (*hook*)

Hooks in the footnotetext command:

```
\NewMirroredHookPair{fntext/before}{fntext/after}
\NewHook{fntext}
\NewHook{fntext/para}
\NewHook{fntext/begin}
\NewHook{fntext/end}
```

5.9 Debugging code

The debugging code is just temporary

```
\bool_new:N        \g_fnote_debug_bool
```

`\DebugFNotesOn`
`\DebugFNotesOff`

```
\cs_new_protected:Npn \DebugFNotesOn  { \bool_gset_true:N  \g_fnote_debug_bool  }
\cs_new_protected:Npn \DebugFNotesOff { \bool_gset_false:N  \g_fnote_debug_bool }
```

(*End of definition for* `\DebugFNotesOn` *and* `\DebugFNotesOff`.)

We log the hooks in the footnote mark command, but only once

```
\cs_new_protected:Npn \__fnote_debug_footnotemark:
  {
    \bool_if:NT \g_fnote_debug_bool
       {
         \hook_log:n {fnmark/before}
         \hook_log:n {fnmark}
         \hook_log:n {fnmark/begin}
         \hook_log:n {fnmark/end}
         \hook_log:n {fnmark/after}
         \cs_gset_eq:NN \__fnote_debug_footnotemark: \prg_do_nothing:
       }
  }
```

Similar for the footnotetext

```
\cs_new_protected:Npn \__fnote_debug_footnotetext:
  {
    \bool_if:NT \g_fnote_debug_bool
       {
         \socket_log:n {fntext/process}
         \socket_log:n {fntext/make}
         \socket_log:n {fntext/begin}
         \socket_log:n {fntext/end}

         \socket_log:n {fntext/mark}
         \socket_log:n {fntext/text}

         \socket_log:n {tagsupport/fnmark}
         \socket_log:n {tagsupport/fntext/begin}
         \socket_log:n {tagsupport/fntext/end}
         \socket_log:n {tagsupport/fntext/mark}
         \socket_log:n {tagsupport/fntext/text}

         \hook_log:n {fntext/before}
         \hook_log:n {fntext}
         \hook_log:n {fntext/para}
         \hook_log:n {fntext/begin}
         \hook_log:n {fntext/end}
         \hook_log:n {fntext/after}
```

Show the info only once (if at all).

```
         \cs_gset_eq:NN \__fnote_debug_footnotetext: \prg_do_nothing:
       }
  }
```

5.10 The new \@footnotemark command

\fnote_footnotemark:

This is the main command which will replace \@footnotemark.

```
\cs_new_protected:Npn \fnote_footnotemark: {
  \__fnote_debug_footnotemark:
%-------
% bibarts
% chextras  --- actually in the wrong place does an \unskip
  \hook_use:n {fnmark/before}
%-------
  \leavevmode
```

```
  \ifhmode
    \edef\@x@sf{\the\spacefactor}
%-------
% bxjsja-minimal.def   --- what they do could be done at ``bibarts''
%                          (a bit less efficient)
% memhfixc.sty
% footmisc.sty
    \hook_use:n {fnmark}
%-------
    \nobreak
  \fi
%-------
% hyperref.sty
  \hook_use:n {fnmark/begin}
%-------
```

The kernel socket for tagging. It picks up `\@makefnmark` as its argument. For link support it should surround the argument with the link socket.

```
  \tag_socket_use:nnn {fnmark}{}{ \socket_use:n{fnmark/link}{\@makefnmark} }
%-------
```

If a footnote mark is placed by its own then it should finish by executing the hook `fnmark/end`, resetting the space factor, and finishing with the hook `fnmark/after`. However, in a complete footnote these actions have to happen only after we have handled the footnote text (e.g., by placing it into an `\insert`). In such a situation `\__fnote_footmark_finish:` below does nothing and the action is carried out later.

```
  \__fnote_footnotemark_finish:
}
```

(*End of definition for* `\fnote_footnotemark:`.)

`\__fnote_footnotemark_default_finish:`
`\__fnote_footnotemark_finish:`

The default definition for `\__fnote_footnotemark_finish:` is called `\__fnote_footnotemark_default_finish:`

```
\cs_new_protected:Npn \__fnote_footnotemark_default_finish: {
% hyperref.sty
% memhfixc.sty  --- could move fnmark/after
% scrlttr2.cls  --- could vanish if footmisc uses a hook
% footmisc.sty
  \UseHook{fnmark/end}
%-------
  \ifhmode
    \spacefactor \@x@sf \relax
  \fi
%
%-------
  \UseHook{fnmark/after}
%-------
}

\cs_new_eq:NN \__fnote_footnotemark_finish: \__fnote_footnotemark_default_finish:
```

(*End of definition for* `\__fnote_footnotemark_default_finish:` *and* `\__fnote_footnotemark_finish:`.)

`tagsupport/fnmark` (*socket*) Not a public socket but reserved for tagging. By default (without tagging) it uses the transparent plug.

```
\NewTaggingSocket{fnmark}{2}
```

\@footnotemark Here we provide the traditional LaTeX 2_ε name in case it is directly used in some legacy class.

```
\cs_set_eq:NN  \@footnotemark \fnote_footnotemark:
```

(*End of definition for* `\@footnotemark`.)

5.11 The new `\@footnotetext` command

`\fnote_footnotetext:n`

```
\cs_new_protected:Npn \fnote_footnotetext:n #1 {
  \__fnote_debug_footnotetext:
%-------
% ./linguex/linguex.sty
  \hook_use:n {fntext/before}
%-------
```

Execute a kernel socket for tagging.

```
  \tag_socket_use:n {fntext/begin}
  \socket_use:nn {fntext/process}
    {
%-------
% resetting baselinestretch ... (could be done further down)
% ./uafthesis/uafthesis.cls
% ./setspace/setspace.sty
% ./footmisc/footmisc.sty (normal)
    \hook_use:n {fntext}
%-------
    \reset@font
    \footnotesize
%-------
% some classes use a different font size, e.g.,
% ./nrc/nrc1.cls  ./nrc/nrc2.cls
% but those could be done in fntext/para instead
%-------
```

In case of sidenotes the next settings are pointless, but as they do not hurt (except for the `\hsize` setting) and are needed for all other cases we make them here and overwrite them for side notes

```
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox
    \floatingpenalty \@MM
    \hsize\columnwidth
    \@parboxrestore
    \UseTaggingSocket{para/restore}
    \parindent 1em     % typical default used in \@makefntext moved up here
    \def\@currentcounter{footnote}
    \protected@edef \@currentlabel { \p@footnote \@thefnmark }
%-------
% for altering para parameters ...
% code for resphilosophica came earlier but it could go here.
% Has the advantage that one can also overwrite \cs{@currentcounter}
% and \cs{@currentlabel} is that is necessary.
%
```

```
% ./resphilosophica/resphilosophica.cls
    \hook_use:n {fntext/para}
%-------
    \color@begingroup
%-------
% fnpara wants to replace \@makefntext{...} and para and side
% option of footmisc etc too ...
% so we make this a socket, because only one action can be active:
%-------
      \socket_use:nn {fntext/make}
        {
%-------
% ./resphilosophica/resphilosophica.cls
%-------
          \socket_use:n {fntext/begin}%
%-------
% bibarts
% fnbreak.sty
          \hook_use:n {fntext/begin}
%-------
          \ignorespaces
          #1
%-------
% bibarts
% fnbreak.sty
          \hook_use:n {fntext/end}
%-------
```

The socket code (by default adding a strut) has to come *after* everything added into the hook above.

```
          \socket_use:n {fntext/end}
        }
      \par
    \color@endgroup
  }
%-------
```

The corresponding kernel hook that ends the tagging structure if tagging is active.

```
  \tag_socket_use:n{fntext/end}
%-------
% ./linguex/linguex.sty
  \hook_use:n {fntext/after}
%-------
}
```

(*End of definition for* `\fnote_footnotetext:n`.)

`fntext/process` (*socket*)

```
\NewSocket    {fntext/process}{1}
\NewSocketPlug{fntext/process}{default}{ \insert\footins {#1} }
\NewSocketPlug{fntext/process}{side}   { \marginpar {#1} }
\AssignSocketPlug{fntext/process}{default}
```

`fntext/make` (*socket*) This socket receives the ⟨*text*⟩ from the `\footnote` or `\footnotetext` and formats it.

```
\NewSocket    {fntext/make}{1}
\NewSocketPlug{fntext/make}{default}{ \@makefntext {#1} }
```

When running several footnotes together as a paragraph some additional work is necessary to unbox the individual footnotes recursively (see TEXbook algorithm in appendix D).

```
\NewSocketPlug{fntext/make}{para}
{
  \setbox\FN@tempboxa\hbox{\@makefntext{#1}}%
  \dp\FN@tempboxa\z@
  \ht\FN@tempboxa
  \dimexpr\wd\FN@tempboxa *%
              \footnotebaselineskip /\columnwidth\relax
  \box\FN@tempboxa
}
\AssignSocketPlug{fntext/make}{default}
```

`fntext/begin` (*socket*) By default adds a strut at the start of the footnote text.

```
\NewSocket    {fntext/begin}{0}
\NewSocketPlug{fntext/begin}{default}{ \rule\z@\footnotesep }
\AssignSocketPlug{fntext/begin}{default}
```

`fntext/end` (*socket*) By default adds a strut at the end of the footnote text unless we are no longer in hmode.

```
\NewSocket    {fntext/end}{0}
\NewSocketPlug{fntext/end}{default}{ \@finalstrut\strutbox }
```

When running several footnotes together as a paragraph some additional glue has to be added between them.

```
\NewSocketPlug{fntext/end}{para}
  {%
            \strut
            \penalty-10\relax
            \hskip\footglue
  }
\AssignSocketPlug{fntext/end}{default}
```

`tagsupport/fntext/begin` (*socket*)
`tagsupport/fntext/end` (*socket*)

Kernel sockets for tagging.

```
\NewTaggingSocket{fntext/begin}{0}
\NewTaggingSocket{fntext/end}{0}
```

Provide the name LaTeX 2_ε is used to and do this unconditionally (no patching of class code if any). This means that if a class provides it own definition that gets lost and if necessary needs to be handled with firstaid (or updating of the class).

```
\AddToHook{begindocument}
  {
    \cs_set_eq:NN \@footnotetext \fnote_footnotetext:n
  }
```

5.12 The new \@makefntext command

\footnotemargin is the logic implemented by footmisc. Perhaps we don't want to do this like that in the kernel but for now I have used this interface unchanged.

```
\newdimen\footnotemargin
\footnotemargin\maxdimen         % no value given

\AtBeginDocument
  {
    \ifdim \footnotemargin=\maxdimen
      \setlength\footnotemargin{1.8em}
    \fi
  }
```

\fnote_makefntext:n

```
\cs_new_protected:Npn \fnote_makefntext:n #1 {
```

Some classes in their redefinition for \@makefntext have placed some paragraph parameters at this point, but those can equally well go into the hook fntext/para. We therefore do not provide a further hook at this point.

```
  \noindent
```

after leaving the vmode we can set the link targets.

```
  \socket_use:n{fntext/mark/link}

  \tag_socket_use:nnn {fntext/mark} {} { \socket_use:n  {fntext/mark} }
  \tag_socket_use:nnn {fntext/text} {} { \socket_use:nn {fntext/text}{#1} }
}
```

(*End of definition for* \fnote_makefntext:n.)

fntext/mark (*socket*) A socket to typeset the mark at the start of a footnote.

```
\NewSocket    {fntext/mark}{0}
```

The default plug implements the logic introduced with the footmisc package.

```
\NewSocketPlug{fntext/mark}{default}{
   \ifdim\footnotemargin>\z@
     \hb@xt@ \footnotemargin{\hss\@makefnmark}
   \else
     \ifdim\footnotemargin=\z@
       \llap{\@makefnmark}
     \else
     \ifdim\footnotemargin=-\maxdimen
          \@makefnmark
       \else
          \llap{\hb@xt@ -\footnotemargin{\@makefnmark\hss}}
       \fi
     \fi
   \fi
}

\AssignSocketPlug{fntext/mark}{default}
```

fntext/text (*socket*) By default this socket does nothing special and simply processes its argument as provided.

```
\NewSocket    {fntext/text}{1}
```

tagsupport/fntext/mark (*socket*) Not a public socket but reserved for tagging. By default they contain the `transparent`
tagsupport/fntext/text (*socket*) and are reassigned if tagging is active.

```
\NewTaggingSocket{fntext/mark}{2}
\NewTaggingSocket{fntext/text}{2}
```

5.12.1 Making documents use the new \@makefntext

If the definition for `\@makefntext` is that of the standard classes then replace it with `\fnote_makefntext:n`, otherwise try to patch the definition used in the class.

Here is the definition the way it is in `classes.dtx`. Notice that (for saving space) there is no space after `em` to terminate the assignment. We need to mimic that, otherwise a test would return false even if the definition has not been modified.

\old@std@class@makefntext

```
\newcommand\old@std@class@makefntext[1]{%
    \parindent 1em%
    \noindent
    \hb@xt@1.8em{\hss\@makefnmark}#1}
```

(*End of definition for* `\old@std@class@makefntext`.)

Here is the messy code for patching. Note that this is only there to help classes along that aren't updated yet so it does some minimal patching to hopefully add kernel configuration hooks in the right place while otherwise leaving the legacy code alone. An updated class would not redefine `\@makefntext` but simply add appropriate code to the provided hooks.

What it does is roughly the following: It looks for a definition of `\@makefntext` of the form

```
{AAA \hbox BBB { CCC } DDD #1 EEE }
```

where "BBB" is something like `to 1em` or similar. It then replaces that with

```
{AAA \UseTaggingSocket{fntext/mark}{}{\hbox BBB { CCC }} DDD
     \UseTaggingSocket{fntext/text}{}{#1} EEE }
```

The patching is not very careful, i.e., it assumes there is only one `#1` in the replacement text and that the first `\hbox` found is the right one to patch. But that is enough to cater for all definitions of `\@makefntext` out there in the TL distribution.

If `\hbox` is not found it tries the same looking for `\hb@xt@` which is what some classes use and if that is not found either it assume that this is a version that uses `\@makefnmark` without surrounding it in a box and if that fails it gives up with an `\ERROR` (which needs to get a proper definition).

```
\tl_new:N \l__fnote_patch_tl
\cs_new_eq:NN \__fnote_tmp:w \ERROR

\cs_new_protected:Npn \__fnote_patch:
  {
    \tl_set:No \l__fnote_patch_tl { \@makefntext { \UseTaggingSocket{fntext/text}{}{##1} } }
    \tl_if_in:NnTF \l__fnote_patch_tl { \hbox }
      { \cs_set_eq:NN \__fnote_tmp:w \__fnote_patch_hbox:w }
      {
        \tl_if_in:NnTF \l__fnote_patch_tl { \hb@xt@ }
          { \cs_set_eq:NN \__fnote_tmp:w \__fnote_patch_hb@xt@:w }
          {
```

Some styles/classes use `\makebox[...][...]` instead of `\hb@xt@` so try to patch those too.

```
            \tl_if_in:NnTF \l__fnote_patch_tl { \makebox }
              { \cs_set_eq:NN \__fnote_tmp:w \__fnote_patch_makebox:w }
              {
                \tl_if_in:NnTF \l__fnote_patch_tl { \@makefnmark }
                  { \cs_set_eq:NN \__fnote_tmp:w \__fnote_patch_@makefnmark:w }
                  { \ERROR
                    \cs_set_eq:NN \__fnote_tmp:w \exp_stop_f: }
              }
          }
      }
    \tl_set:Nf \l__fnote_patch_tl
      { \exp_after:wN \__fnote_tmp:w \l__fnote_patch_tl }
    \cs_set:Npn \__fnote_tmp:w { \long \def \@makefntext ####1 }
    \exp_after:wN \__fnote_tmp:w \exp_after:wN { \l__fnote_patch_tl }
  }
```

If `\@makefntext` contains `\hbox` then grab "AAA" as `#1` and "BBB" (up to the open `{`) and return it as

```
AAA  \@makefntext@processX { \hbox BBB }
```

```
\cs_new:Npn \__fnote_patch_hbox:w #1 \hbox #2 #
  { \exp_stop_f: #1 \@makefntext@processX { \hbox #2 } }
```

Same for the other cases.

```
\cs_new:Npn \__fnote_patch_hb@xt@:w #1 \hb@xt@ #2 #
  { \exp_stop_f: #1 \@makefntext@processX { \hb@xt@ #2 } }
\cs_new:Npn \__fnote_patch_makebox:w #1 \makebox #2 #
  { \exp_stop_f: #1 \@makefntext@processX { \makebox #2 } }
```

If the definition contains neither `\hbox`, `\hb@xt@` nor `\makebox`, we see if it contains `\@makefnmark` and if so put the socket before that.

```
\cs_new:Npn \__fnote_patch_@makefnmark:w #1 \@makefnmark
  { \exp_stop_f: #1 \@makefntext@processX { \use:n } { \@makefnmark } }
```

The code provided by Bruno above expects 2 arguments but we need a different structure so this is a simple reshuffling. Would be better if we can patch the right structure in directly, but I'm not a patch person, so this is the simple way out for now:

```
\cs_new:Npn \@makefntext@processX #1#2
  {
    \socket_use:n{fntext/mark/link}
    \UseTaggingSocket{fntext/mark}{}{#1{#2}}
  }
```

At `\begin{document}` check if the current definition is that of the standard classes and if so replace it by `\fnote_makefntext:n` otherwise try and patch the definition made by some class or package using the approach above.

```
\AddToHook{begindocument}
  {
    \cs_if_eq:NNF\@makefntext \fnote_makefntext:n
     {
       \cs_if_eq:NNTF \@makefntext \old@std@class@makefntext
         {
```

```
            \cs_set_eq:NN \@makefntext \fnote_makefntext:n
          }
          {
```

If `\@makefntext` contains the definition from footmisc we do nothing, otherwise we try to patch.

```
            \cs_if_eq:NNF \@makefntext \footmisc@hang@makefntext
                          { \__fnote_patch: }
          }
      }
  }


% possibly add the following to check for multiple \hbox in
% the definition:
%
% \seq_set_split:NnV \l__fnote_patch_seq { \hbox } \l__fnote_patch_tl
% \int_compare:nT { \seq_count:N \l__fnote_patch_seq } > 2 \ERROR
%
```

5.13 Document-level commands

`\footnotetext`

```
\DeclareDocumentCommand\footnotetext {o+m}
  {
    \fnote_set_fnmark:nn {#1} \@mpfn
    \@footnotetext {#2}
  }
```

(*End of definition for* `\footnotetext`.)

`\footnote`

```
\DeclareDocumentCommand\footnote {o+m}
  {
    \fnote_step_fnmark:nn {#1} \@mpfn
    \cs_set_eq:NN \__fnote_footnotemark_finish: \prg_do_nothing:
    \@footnotemark
    \cs_set_eq:NN \__fnote_footnotemark_finish: \__fnote_footnotemark_default_finish:
    \@footnotetext {#2}
    \__fnote_footnotemark_finish:
  }
```

(*End of definition for* `\footnote`.)

`\footnotemark`

```
\DeclareDocumentCommand\footnotemark {o}
  {
    \fnote_step_fnmark:nn {#1} { footnote }
    \@footnotemark
  }
```

(*End of definition for* `\footnotemark`.)

\footref

\footref used the starred \ref in \@thefnmark as the linking is handled by the tagging code inside the \@footnotemark. \footref should not try to link to its related note automatically but should instead use the label. This is passed to \@footnotemark through \l__fnote_currentlabel_tl.

```
\DeclareDocumentCommand\footref {m}
  {
    \begingroup
      \unrestored@protected@xdef\@thefnmark{\ref*{#1}}%
    \endgroup
    \bool_set_false:N  \l__fnote_autodetect_bool
    \tl_set:Nn \l__fnote_currentlabel_tl {#1}
    \@footnotemark
    \bool_set_true:N  \l__fnote_autodetect_bool
  }
```

(*End of definition for* \footref.)

5.14 Firstaid for packages and classes

5.15 Kernel patches

Tagging of footnotes in minipages require a change in the minipage commands We define at first a local configuration command for minipage footnotes.

```
\NewSocketPlug{fntext/process}{mp}
 {
   \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins
    #1
    }
 }
```

5.15.1 memoir

The memoir class redefines various internal commands to inject its hooks and additional code. The following reinstates the kernel command and so probably breaks various options of memoir, but without the changes it errors anyway. The footmisc package should be used to change for example to para footnotes.

```
\AddToHook{class/memoir/before}
  { \let\new@std@class@makecol\@makecol }
\AddToHook{class/memoir/after}
  {
    \cs_set_eq:NN  \@footnotemark \fnote_footnotemark:
    \cs_set_eq:NN  \@makefntext\old@std@class@makefntext
    \cs_set_eq:NN  \@makecol\new@std@class@makecol
  }
```

5.15.2 setspace

It should not overwrite it any longer but use a hook, so for now we do just that here.

```
\AddToHook{package/setspace/after}
   {\let \@footnotetext \fnote_footnotetext:n
    \AddToHook{fntext}[setspace]{\let\baselinestretch\setspace@singlespace}}
```

5.15.3 hyperref

hyperref has a hook which allows to disable its footnote related patches. As we will handle links directly in the code this is used.

```
\def\hyper@nopatch@footnote{}
```

We use the hyperref commands for now for links. To avoid to have to test for hyperref we provide dummies. TODO consider to use specials to get similar spacing.

```
\AtBeginDocument
  {
   \providecommand\hyper@linkstart{\@gobbletwo}
   \providecommand\hyper@linkend{\@empty}
  }
```

It must be possible to suppress the hyperlinking, both locally and globally. hyperref should set the boolean `\l_fnote_link_bool`. For now we test for the hyperref boolean (so it can be suppressed only globally).

```
\AtBeginDocument
 {
   \@ifpackageloaded{hyperref}
     {
       \AssignSocketPlug{fnmark/link}{link}
       \AssignSocketPlug{fntext/mark/link}{link}
       \legacy_if:nF{Hy@hyperfootnotes}{\bool_set_false:N \l_fnote_link_bool}
     }
     {
       \bool_set_false:N \l_fnote_link_bool
     }
 }
```

5.16 Tagging and hyperlink code

`\g_fnote_id_int`

For links we need an unique id to identify footnote marks and footnotes. It is increased both for the mark in the text and the note below (similar to the structure numbers). We make it public to allow the implementation of other footnote commands, see the test-tnote-setup test. The int is increased in the fntext/before hook, so that the following tagging socket can pick it up. TODO: check if the hook is the right place or if this should be static code.

```
\int_new:N \g_fnote_id_int
\AddToHook{fntext/before}{\int_gincr:N\g_fnote_id_int}
```

5.16.1 Rolemap for structure tags

We use role-mapping to get more speaking names in the PDF and so ease debugging. These names are already provided by tagpdf directly.

5.16.2 Extending the label system

For \footref and (perhaps later for labeled footnotes) we must extend the label system. Beside the normal values we also need the structure number and id of the note. We use the in-built label hook to record properties. We define two suitable properties: the one stores the structure number as stored in \l__fnote_currentstruct_tl, the other the id.

```
\property_new:nnnn {fnote/struct}{now}{1}{\l__fnote_currentstruct_tl}
\property_new:nnnn {fnote/id}{now}{1}{\int_use:N\g_fnote_id_int}
```

We add a hook to the label hook. By default it does nothing

__fnote_label_hook:e

```
\cs_new_protected:Npn \__fnote_label_hook:e #1 {}
\AddToHookWithArguments{label}{ \__fnote_label_hook:e{#1}}
```

(*End of definition for* __fnote_label_hook:e.)

Inside a footnotetext we change the hook to store the structure number and id too. The name of label is provided as argument in the label hook.

```
\AddToHook{fntext/begin}
 {
   \cs_set_protected:Npn \__fnote_label_hook:e #1
    {
      \property_record:ee {__fnote/#1} {fnote/struct,fnote/id}
    }
 }
```

5.16.3 Storing and retrieving reference data

To establish the connection between a mark and a note the mark has to store its representation, and the note has to analyse the stored representations to get the structure numbers of its mark. This is done with the public function to allow similar systems (e.g. tabular notes, other footnote series) to make use of this.

\fnote_class_new:nn

This sets up a new footnote type, the first argument is the name, the second is meant for options. Currently it does nothing at all. It is not necessary to set up every footnote command as its own type!

```
\cs_new_protected:Npn \fnote_class_new:nn #1 #2 % #1 name, #2 options
 {
   \prop_new:c { g__fnote_currentmarks_struct_#1_prop }
   \prop_new:c { g__fnote_currentmarks_id_#1_prop }
 }

\fnote_class_new:nn {default}{}
```

(*End of definition for* \fnote_class_new:nn. *This function is documented on page* 9.)

\fnote_mark_id_gput:nn
\fnote_mark_struct_gput:nn

This commands takes as argument the representation of the mark, e.g., \@thefnmark and the type (typically default should work).

```
\cs_new_protected:Npn \fnote_mark_struct_gput:nn #1 #2 % #1 the representation of the mark,
 {
   \prop_gput:cen { g__fnote_currentmarks_struct_#2_prop }
     { \tag_get:n{struct_num} }
     { #1 }
```

```
 }
\cs_new_protected:Npn \fnote_mark_id_gput:nn #1 #2 % #1 the representation of the mark, #2 t
 {
   \prop_gput:cen { g__fnote_currentmarks_id_#2 _prop }
     { \int_use:N\g_fnote_id_int }
     { #1 }
 }
\cs_generate_variant:Nn \fnote_mark_struct_gput:nn {no,oo}
\cs_generate_variant:Nn \fnote_mark_id_gput:nn {no,oo}
```

(*End of definition for* `\fnote_mark_id_gput:nn` *and* `\fnote_mark_struct_gput:nn`*. These functions are documented on page* ??.)

`\fnote_mark_gpop_all:nnnN` This command takes as first argument either id or struct, then representation of the mark (e.g. the content of`\@thefnmark`), the class (typically `default` should work) and a sequence into which every key in the property is stored that has the same value as the mark. The sequence is cleared first.

```
\cs_new_protected:Npn \fnote_mark_gpop_all:nnnN #1 #2 #3 #4
  {
    \seq_clear:N #4
    \prop_set_eq:Nc     \l__fnote_tmpa_prop { g__fnote_currentmarks_#1_#3_prop }
    \prop_map_inline:Nn \l__fnote_tmpa_prop
      {
        \tl_if_eq:nnT {#2} { ##2 }
          {
```

store the key (struct num or id) in the seq

```
            \seq_put_right:Nn #4 { ##1 }
```

remove entry as used from the global prop

```
            \prop_gremove:cn { g__fnote_currentmarks_#1_#3_prop } {##1}
          }
      }
  }
\cs_generate_variant:Nn\fnote_mark_gpop_all:nnnN  {nooN}
```

(*End of definition for* `\fnote_mark_gpop_all:nnnN`*. This function is documented on page 10.*)

5.16.4 Enabling tagging and links for the mark command

`fnmark/link` (*socket*) A socket that contains the code to surround the mark in the text with a link. It should be used in the second argument of the tagging socket as we normally want the link to be inside the Lbl structure.

```
\NewSocket{fnmark/link}{1}
```

`link` (*plug*) To handle the link around the mark we define a plug for the socket fnmark/link

```
\NewSocketPlug{fnmark/link}{link}
 {
   \int_gincr:N\g_fnote_id_int
   \bool_if:NTF \l__fnote_autodetect_bool
     {
        \fnote_mark_id_gput:oo {\@thefnmark}{\l_fnote_type_tl}
        \tl_set:Ne \l__fnote_linktarget_tl {footnote*.\int_use:N\g_fnote_id_int}
     }
     {
```

```
      \tl_set:Ne \l__fnote_linktarget_tl
        {footnote*.\property_ref:ee {__fnote/\l__fnote_currentlabel_tl} {fnote/id}}
    }
   \bool_if:NTF \l_fnote_link_bool
    {
      \exp_args:No
         \hyper@linkstart
         { \l_fnote_link_type_tl }
         { \l__fnote_linktarget_tl }
         #1
         \hyper@linkend
     }
     { #1 }
 }
```

FEMark (*plug*)

To handle the mark in the text, we define a special plug for the socket `tagsupport/fnmark` that receives `\@makefntext` as its argument. At this time `\@thefnmark` is already set.

```
\NewTaggingSocketPlug{fnmark}{FEMark}
 {
```

End an open mc and start the structure.

```
   \tag_mc_end_push:
   \tag_struct_begin:n { tag=footnotemark }
```

The associated note is either auto detected or given by the user. If there is no autodetecting we need some id, currently it is called `\l__fnote_currentlabel_tl`. the Ref is set by looking at the label value. We must also add the current structure number to the `/Ref` of the FEnote. Both must be delayed as we don't know if the objects of the FEnote and the mark have already been created.

```
    \bool_if:NF \l__fnote_autodetect_bool
     {
       \hook_gput_code:nne {tagpdf/finish/before} {tagpdf/footnote}
        {
          \exp_not:N\fnote_gput_refs:ee
           { \tag_get:n{struct_num} }
           { \property_ref:ee{ __fnote/\l__fnote_currentlabel_tl } {fnote/struct} }
        }
     }
```

And now the actual content

```
    \tag_mc_begin:n{tag=Lbl}
```

For the auto detecting we store the struct num and `\@thefnmark` inside a prop TODO: this should be usable for other footnote types which means the name of the prop shouldn't be fix.

```
    \bool_if:NT \l__fnote_autodetect_bool
     {
       \fnote_mark_struct_gput:oo {\@thefnmark}{\l_fnote_type_tl}
     }
```

The target in the footnote will use the current id. This part is needed even if tagging is deactivated.

```
    #2
    \tag_mc_end:
```

```
    \tag_struct_end:
    \tag_mc_begin_pop:n{}
  }
```

At last assign the plug:

```
\AssignTaggingSocketPlug{fnmark}{FEMark}
```

5.16.5 The footnote text

We need a public command to append values to the Ref keys

__fnote_gput_ref:nn
\fnote_gput_refs:nn
\fnote_gput_refs:ee

```
\cs_new_protected:Npn \__fnote_gput_ref:nn #1 #2 %#1 the structure number receiving the ref
  {
    \tag_struct_gput:nnn {#1}{ref_num}{#2}
  }
\cs_new_protected:Npn \fnote_gput_refs:nn #1 #2 % pair of structure numbers
  {
    \__fnote_gput_ref:nn {#1}{#2}
    \__fnote_gput_ref:nn {#2}{#1}
  }
\cs_generate_variant:Nn \fnote_gput_refs:nn {ee}
```

(*End of definition for* `\__fnote_gput_ref:nn` *and* `\fnote_gput_refs:nn`.)

kernel hooks for tagging this sets the structure around the whole text

FENote (*plug*)

```
\NewTaggingSocketPlug{fntext/begin}{FENote}
  {
```

The id has been increased in the previous fntext/before hook.

```
    \tag_mc_end_push:
```

test if a footnote is allowed, if not move up to the next sect or the document structure.

```
    \tag_check_child:nnTF {FENote}{pdf2}
      {
        \tag_struct_begin:n { tag=footnote }
      }
      {
        \tag_struct_begin:n
          {
            tag=footnote,
```

We add 0 for now to ensure to get a number even if the sec code is not loaded or if the seq is empty (which it shouldn't unless there is an coding error)

```
            parent=\int_max:nn{2}{\tag_get:n{current_Sect}+0}
          }
      }
```

Store the current structure number for labels.

```
    \tl_set:Ne \l__fnote_currentstruct_tl { \tag_get:n{struct_num} }
```

After we have opened the structure we can use the `struct num` to try to detect the connected marks. As with the marks we assume that sometimes no auto detection is done.

```
    \bool_if:NTF \l__fnote_autodetect_bool
      {
```

Find open marks with identical `\@thefnmark`:

```
      \fnote_mark_gpop_all:nooN {struct}{ \@thefnmark }{ \l_fnote_type_tl } \l__fnote_curr
```

Then we store the object numbers of the marks in the /Ref of the FENote structure and the number of the FEnote into the marks structure:

```
    \seq_map_inline:Nn \l__fnote_currentrefs_seq
      {
        \fnote_gput_refs:ee {##1}{ \l__fnote_currentstruct_tl }
      }
   }
```

If no auto detection is done

```
   {%no auto

   }
}
```

This finish the setup of the tagging structure.

`FENote` (*plug*) This is the plug for the socket at the end of the structure.

```
\NewTaggingSocketPlug{fntext/end}{FENote}
  {
    \tag_struct_end:
    \tag_mc_begin_pop:n{}
  }
```

At last assign the plugs:

```
\AssignTaggingSocketPlug{fntext/begin}{FENote}
\AssignTaggingSocketPlug{fntext/end}{FENote}
```

`fntext/mark/link` (*socket*) This socket adds targets for links before the mark of the footnote in the note text. It should be used in hmode.

```
\NewSocket{fntext/mark/link}{0}
```

`link` (*plug*) We create a link target for every related mark. The name is `footnote*.`⟨*id*⟩. We also add a link target for the current id (for `\footref`). TODO: perhaps it should be always active.

```
\NewSocketPlug{fntext/mark/link}{link}
  {
    \fnote_mark_gpop_all:nooN {id}{ \@thefnmark }{ \l_fnote_type_tl } \l__fnote_currentrefs_
    \seq_map_inline:Nn\l__fnote_currentrefs_seq {\MakeLinkTarget*{footnote*.##1}}
    \MakeLinkTarget*{footnote*.\int_use:N\g_fnote_id_int}
  }
```

The kernel socket `tagsupport/fntext/mark` is responsible for tagging the mark in the note. We use it to surround the mark with the needed tagging commands.

TODO check if additional kernel configuration points are needed. If yes, what about the paragraph start and the paratagging??

`FENoteLbl` (*plug*) This plug creates the label in the note on the bottom. It also adds link targets for the hyperlinking.

```
\NewTaggingSocketPlug{fntext/mark}{FENoteLbl}
  {
    \tag_mc_end_push:
```

Now we add the tagging commands. We move the structure of the label to the begin of the footnote structure.

```
      \tag_struct_begin:n { tag=footnotelabel,parent=\l__fnote_currentstruct_tl,firstkid }
       \tag_mc_begin:n { tag=Lbl }
        #2
       \tag_mc_end:
      \tag_struct_end:
     \tag_mc_begin_pop:n{}
   }
\AssignSocketPlug{tagsupport/fntext/mark}{FENoteLbl}
```

`FENotetext` (*plug*)

This plug is for the kernel socket `tagsupport/fntext/text` around the actual note text when doing tagging. Currently it only adds an MC chunk.
TODO Should it set a mc or could it rely on the content?

```
\NewTaggingSocketPlug{fntext/text}{FENotetext}
  {
    \tag_mc_end_push:
    \tag_mc_begin:n{}
    #2
    \tag_mc_end:
    \tag_mc_begin_pop:n{}
  }
\AssignTaggingSocketPlug{fntext/text}{FENotetext}
```

```
\ExplSyntaxOff
⟨/kernel⟩
⟨@@=⟩
```

6 Reimplementing the **footmisc** package

```
⟨*footmisc⟩
%%
%% Copyright (c) 1995-2011 Robin Fairbairns
%% Copyright (c) 2018-2023 Robin Fairbairns, Frank Mittelbach
%%
%% This file is part of the `latex-lab Bundle'.
%% -------------------------------------------
%%
%% It may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3c
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%    https://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of LaTeX
%% version 2008 or later.
%%
%% This work has the LPPL maintenance status 'maintained'.
%%
\NeedsTeXFormat{LaTeX2e}
```

```
\providecommand\DeclareRelease[3]{}
\providecommand\DeclareCurrentRelease[2]{}

\DeclareRelease{v5}{2011-06-06}{footmisc-2011-06-06.sty}
\DeclareCurrentRelease{}{2022-02-14}
\ProvidesPackage{latex-lab-footmisc}%
        [2025/02/20 v6.0e
     a miscellany of footnote facilities -- latex-lab version%
                   ]

\NeedsTeXFormat{LaTeX2e}[2020/10/01]
\newtoks\FN@temptoken
\providecommand\protected@writeaux{%
  \protected@write\@auxout
}
\def\l@advance@macro{\@@dvance@macro\edef}
\def\@@dvance@macro#1#2#3{\expandafter\@tempcnta#2\relax
  \advance\@tempcnta#3\relax
  #1#2{\the\@tempcnta}%
}
\let\@advance@macro\l@advance@macro
\DeclareOption{symbol}{\renewcommand\thefootnote{\fnsymbol{footnote}}}
\newif\ifFN@robust \FN@robustfalse
\DeclareOption{symbol*}{%
  \renewcommand\thefootnote{\@fnsymbol\c@footnote}%
  \FN@robusttrue
  \AtEndOfPackage{\setfnsymbol{lamport*-robust}}%
}
\newif\ifFN@para  \FN@parafalse
\DeclareOption{para}{%
```

Options are executed in the order of declaration, thus no point in checking for side option as footmisc did in the past

```
%    \PackageError{footmisc}{Option "\CurrentOption" incompatible with
%                            option "side"}%
%                 {I shall ignore "\CurrentOption"}%
  \FN@paratrue
  \setlength\footnotemargin{-\maxdimen}    % default when para is used
}
\DeclareOption{side}{\ifFN@para
    \PackageError{footmisc}{Option "\CurrentOption" incompatible with
                            option "para"}%
                {I shall ignore "\CurrentOption"}%
  \else
    \AddToHook{fntext/para}{%
      \hsize\marginparwidth     % correct the default \hsize
      \footnotesep\z@           % don't add a default separation
    }
    \AssignSocketPlug{fntext/process}{side}
%   \AssignSocketPlug{fntext/make}{default}
    \AssignSocketPlug{fntext/begin}{noop}
    \AssignSocketPlug{fntext/end}{noop}
  \fi
}
```

```
\let\footnotelayout\@empty
\DeclareOption{ragged}{%
  \@ifundefined{RaggedRight}%
    {\renewcommand\footnotelayout{\linepenalty50 \raggedright}}%
    {\renewcommand\footnotelayout{\linepenalty50 \RaggedRight}}%
}
\newif\ifFN@perpage
\FN@perpagefalse
\DeclareOption{perpage}{%
  \FN@perpagetrue
}
\newif\ifFN@fixskip      \FN@fixskipfalse

\let\FN@bottomcases\thr@@
\newif\ifFN@abovefloats  \FN@abovefloatstrue
\DeclareOption{bottom}{%
  \let\FN@bottomcases\@ne
  \FN@abovefloatsfalse
  \FN@fixskiptrue
}
\DeclareOption{bottomfloats}{%
  \let\FN@bottomcases\tw@
  \FN@abovefloatstrue \FN@fixskiptrue
}
\DeclareOption{abovefloats}{\FN@abovefloatstrue  \FN@fixskiptrue}
\DeclareOption{belowfloats}{\FN@abovefloatsfalse \FN@fixskiptrue}
\DeclareOption{marginal}{%
  \footnotemargin-0.8em\relax
}
\DeclareOption{flushmargin}{%
  \footnotemarginOpt\relax
}
\newif\ifFN@hangfoot  \FN@hangfootfalse
\DeclareOption{hang}{%
  \FN@hangfoottrue
}
\newcommand*\hangfootparskip{0.5\baselineskip}
\newcommand*\hangfootparindent{0em}%
\DeclareOption{norule}{%
  \renewcommand\footnoterule{}%
  \advance\skip\footins 4\p@\@plus2\p@\relax
}
\DeclareOption{splitrule}{%
  \gdef\split@prev{0}
  \let\pagefootnoterule\footnoterule
  \let\mpfootnoterule\footnoterule
  \def\splitfootnoterule{\kern-3\p@ \hrule \kern2.6\p@}
  \def\footnoterule{\relax
    \ifx \@listdepth\@mplistdepth
      \mpfootnoterule
    \else
      \ifnum\split@prev=\z@
        \pagefootnoterule
      \else
```

```
        \splitfootnoterule
      \fi
      \xdef\split@prev{\the\insertpenalties}%
    \fi
  }%
}
\newif\ifFN@stablefootnote  \FN@stablefootnotefalse
\DeclareOption{stable}{\FN@stablefootnotetrue}
\newif\ifFN@multiplefootnote  \FN@multiplefootnotefalse
\DeclareOption{multiple}{\FN@multiplefootnotetrue}
\ProcessOptions
```

Footnote box layout for para footnotes; this would also be the hook to support dblfootnotes (from the `dblfnote` package if we integrate that).

```
\ifFN@para
  \NewSocketPlug{build/column/footnotes}{para}{%
     \global\setbox\footins\vbox{\FN@makefootnoteparagraph}%
    }
  \AssignSocketPlug{build/column/footnotes}{para}
\fi
\ifFN@fixskip
  \def\@outputbox@removebskip{%
    \ifx\@textbottom\relax \else
      \@outputbox@append{%
        \@tempskipa\lastskip
        \ifnum \gluestretchorder\@tempskipa>\z@
          \vskip-\@tempskipa
          \xdef\@outputbox@reinsertbskip
              {\noexpand\@outputbox@append{\vskip\the\@tempskipa}}%
        \else
          \global\let\@outputbox@reinsertbskip\relax
        \fi
      }%
   \fi
  }
\let\@outputbox@reinsertbskip\relax
\else
  \let\@outputbox@removebskip \relax
  \let\@outputbox@reinsertbskip\relax
\fi
\ifcase \FN@bottomcases\relax
  \ERROR
\or %1  bottom option
  \ifFN@abovefloats
    \AssignSocketPlug {build/column/outputbox}{space-footnotes-floats}
  \else
    \AssignSocketPlug {build/column/outputbox}{floats-space-footnotes}
  \fi
\or %2  bottomfloats
  \ifFN@abovefloats
    \AssignSocketPlug {build/column/outputbox}{footnotes-space-floats}
  \else
    \AssignSocketPlug {build/column/outputbox}{space-floats-footnotes}
```

```
  \fi
\or %3 neither bottom nor bottomfloats
  \ifFN@abovefloats
    \AssignSocketPlug {build/column/outputbox}{footnotes-floats}
  \else
    \AssignSocketPlug {build/column/outputbox}{floats-footnotes}
  \fi
\else
  \ERROR
\fi

% next can be dropped when cleaned up
\newif\ifFN@setspace
\@ifpackageloaded{setspace}%
 {%
   \FN@setspacetrue
   \@ifclassloaded{memoir}%
     {%
       \AddToHook{fntext}{\let\baselinestretch\m@m@singlespace}%
       \let\FN@baselinestretch\m@m@singlespace
     }%
     {%
%       \AddToHook{fntext}{\let\baselinestretch\setspace@singlespace}%
       \let\FN@baselinestretch\setspace@singlespace
     }%
 }%
 {%
   \FN@setspacefalse
 }



\ifFN@para

% \AssignSocketPlug{fntext/process}{default}
  \AssignSocketPlug{fntext/make}{para}
  \AssignSocketPlug{fntext/begin}{noop}
  \AssignSocketPlug{fntext/end}{para}

\fi



\ifFN@para
  \let\FN@tempboxa\@tempboxa
  \newbox\FN@tempboxb
  \newbox\FN@tempboxc
  \newskip\footglue \footglue=1em plus.3em minus.3em

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  \newdimen\footnotebaselineskip

  % establish late:

```

```
\AddToHook{begindocument/before} {%
  {%
    \footnotesize
    \global\footnotebaselineskip=\normalbaselineskip
  }%
}
```

The coding is based on David Kastrup's improvement to Don Knuth's original implementation. You find in the TEXbook if you own the latest edition.

```
  \long\def\FN@makefootnoteparagraph{%
    \FN@setfootnoteparawidth
    \@parboxrestore
    \baselineskip=\footnotebaselineskip
    \unvbox\footins \FN@removehboxes
    \RawParEnd
  }
  \def\FN@removehboxes{\setbox\FN@tempboxa\lastbox
    \ifhbox\FN@tempboxa{\FN@removehboxes}%
      \unhbox\FN@tempboxa
    \else
      \RawNoindent
      \rule\z@\footnotesep
    \fi
  }
\fi


\@ifpackageloaded{multicol}
  {\def\FN@setfootnoteparawidth
    {\hsize\ifnum\doublecol@number>\@ne
                  \textwidth
            \else \columnwidth \fi}}
  {\def\FN@setfootnoteparawidth{\hsize\columnwidth}}

\ifFN@perpage
  \RequirePackage{perpage}
  \MakePerPage{footnote}
```

Fix a bug in perpage ...

```
  \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne
    \stepcounter{#1}%
    \pp@fix@MakePerPage{#1}%
  }
  \def\pp@fix@MakePerPage#1{%
      \ifnum \value{#1}>\z@
        \addtocounter{#1}\m@ne\fi
  }
```

The above code may look a bit odd: the `\stepcounter` sets the counter to zero and then we alter it if it is not zero. The reason is that `\stepcounter` resets other counters and when perpage is loaded this results in updating counters on the reset list to 1 (or to a higher starting value if `\MakePerPage` is used with an optional argument, which is precisely the problem here). By subtracting 1 in that case we set it back to 1 lower than the starting value.

But to make this fully work we also need to update a support command in perpage:

```
  \def\pp@cl@end@iii\stepcounter#1\pp@fix@MakePerPage#2{}
\fi


\ifFN@para

% This can use the default interface, except that a negative value for
% \footnotemargin makes little sense, so we test for this and warn if
% necessary. But -\maxdimen is ok again, so would need to be a little bit more elaborate.
%

%\AddToHook{fntext/para}{
%  \ifdim \footnotemargin >\z@ \else
%    \PackageWarningNoline{footmisc}{Option 'para' needs positive \noexpand\footnotemargin}%
%    \footnotemargin 1.8em\relax
%  \fi
%}


  \AddToHook{fntext/begin}{\nobreak \hspace{.2em}}

\else

  \ifFN@hangfoot
    \long\def\footmisc@hang@makefntext#1{%
      \bgroup
        \SuspendTagging{footmisc}%
        \setbox\@tempboxa\hbox{%
          \ifdim\footnotemargin>\z@
            \hb@xt@\footnotemargin{\@makefnmark\hss}%
          \else
            \@makefnmark
          \fi
        }%
        \leftmargin\wd\@tempboxa
        \rightmargin\z@
        \linewidth \columnwidth
        \advance \linewidth -\leftmargin
        \parshape \@ne \leftmargin \linewidth
        \footnotesize
        \parskip\hangfootparskip\relax
```

The setting of `\parindent` has to happen prior to calling `\leavevmode` (which happens below to support tagging). Originally, this was done later so was moved here where we haven't started hmode yet.

```
        \parindent\hangfootparindent\relax
        \@setpar{{\@@par}}%
        \ResumeTagging{footmisc}%
        \leavevmode
```

after leaving the vmode we can set the link targets. They should be in the footnote structure.

```
        \UseSocket{fntext/mark/link}%
```

Typesetting the mark twice means that one can't have any material inside that gets unhappy in that case. That shouldn't be a problem, but perhaps we have to come up with a more elaborate solution in the end.

```
        \UseTaggingSocket{fntext/mark}{}%
                  {\llap{%
                    \ifdim\footnotemargin>\z@
                      \hb@xt@\footnotemargin{\@makefnmark\hss}%
                    \else
                      \@makefnmark
                    \fi
                  }}%
        \footnotelayout#1%
        \par
      \egroup
    }
```

Defined in a roundabout way so that we can test for it when patching classes that are not updated.

```
    \let \@makefntext \footmisc@hang@makefntext

 \else

% This is now using the default interface:
%
% \long\def\@makefntext#1{%
%      \parindent1em
%      \noindent
%      \ifdim\footnotemargin>\z@
%        \hb@xt@ \footnotemargin{\hss\@makefnmark}%
%      \else
%        \ifdim\footnotemargin=\z@
%          \llap{\@makefnmark}%
%        \else
%          \llap{\hb@xt@ -\footnotemargin{\@makefnmark\hss}}%
%        \fi
%      \fi
%    \footnotelayout#1%
%  }

 \fi
\fi




\ifFN@multiplefootnote
  \providecommand*{\multiplefootnotemarker}{3sp}
```

We tag the separator as artifact

TODO: *why is this done with* `\providecommand`*?*

```
  \ExplSyntaxOn
  \providecommand*{\multfootsep}{\tag_mc_end_push:\tag_mc_begin:n{artifact},\tag_mc_end:\tag
  \ExplSyntaxOff
```

```
  \AddToHook{fnmark}      {\FN@mf@check}
  \AddToHook{fnmark/end}  {\FN@mf@prepare}
%
  \def\FN@mf@prepare{%
    \kern-\multiplefootnotemarker
    \kern\multiplefootnotemarker\relax
  }
  \def\FN@mf@check{%
    \ifdim\lastkern=\multiplefootnotemarker\relax
%?? is that necessary or even correct ??
      \edef\@x@sf{\the\spacefactor}%
%?? shouldn't that be 2 unkerns ?? (none would also be ok)
      \unkern  % new
      \unkern
      \textsuperscript{\multfootsep}%
      \spacefactor\@x@sf\relax
    \fi
  }
\else
  \let\FN@mf@prepare\relax
\fi
\ifFN@stablefootnote
\let\FN@sf@@footnote\footnote
\def\footnote{\ifx\protect\@typeset@protect
    \expandafter\FN@sf@@footnote
  \else
    \expandafter\FN@sf@gobble@opt
  \fi
}
\edef\FN@sf@gobble@opt{\noexpand\protect
  \expandafter\noexpand\csname FN@sf@gobble@opt \endcsname}
\expandafter\def\csname FN@sf@gobble@opt \endcsname{%
  \@ifnextchar[%]
    \FN@sf@gobble@twobracket
    \@gobble
}
\def\FN@sf@gobble@twobracket[#1]#2{}
\let\FN@sf@@footnotemark\footnotemark
\def\footnotemark{\ifx\protect\@typeset@protect
    \expandafter\FN@sf@@footnotemark
  \else
    \expandafter\FN@sf@gobble@optonly
  \fi
}
\edef\FN@sf@gobble@optonly{\noexpand\protect
  \expandafter\noexpand\csname FN@sf@gobble@optonly \endcsname}
\expandafter\def\csname FN@sf@gobble@optonly \endcsname{%
  \@ifnextchar[%]
    \FN@sf@gobble@bracket
    {}%
}
\def\FN@sf@gobble@bracket[#1]{}
\fi
\newcommand\setfnsymbol[1]{%
```

```
  \@bsphack
  \@ifundefined{FN@fnsymbol@#1}%
  {%
    \PackageError{footmisc}{Symbol style "#1" not known}%
    \@eha
  }{%
    \expandafter\let\expandafter\@fnsymbol\csname
                          FN@fnsymbol@#1\endcsname
  }%
  \@esphack
}
\let\FN@fnsymbol@lamport\@fnsymbol
\newif\if@tempswb
\DeclareDocumentCommand\DefineFNsymbols {smO{text}m}{%
  \expandafter\ifx\csname FN@fnsymbol@#2\endcsname\relax
    \PackageInfo{footmisc}{Declaring symbol style #2}%
  \else
    \PackageWarning{footmisc}{Redeclaring symbol style #2}%
  \fi
  \toks@{}%
  \def\@tempb{\end}%
  \FN@build@symboldef#4\end
  \def\@tempc{math}%
  \def\@tempd{#3}%
  \expandafter\xdef\csname FN@fnsymbol@#2\endcsname##1{%
    \ifx\@tempc\@tempd
      \noexpand\ensuremath
    \else
      \noexpand\nfss@text
    \fi
    {%
      \noexpand\ifcase##1%
      \the\toks@
      \noexpand\else
      \IfBooleanTF#1{\noexpand\@ctrerr}%
        {\noexpand\FN@orange##1}%
      \noexpand\fi
    }%
  }%
}
\def\FN@build@symboldef#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\@tempb
  \else
    \toks@\expandafter{\the\toks@\or#1}%
    \expandafter\FN@build@symboldef
  \fi
}
\DeclareDocumentCommand\DefineFNsymbolsTM {smm}{%
  \expandafter\ifx\csname FN@fnsymbol@#2\endcsname\relax
    \PackageInfo{footmisc}{Declaring symbol style #2}%
  \else
    \PackageWarning{footmisc}{Redeclaring symbol style #2}%
  \fi
```

```
  \toks@{}%
  \def\@tempb{\end}%
  \FN@build@symboldefTM#3\end\@null
  \expandafter\xdef\csname FN@fnsymbol@#2\endcsname##1{%
    \noexpand\ifcase##1%
      \the\toks@
    \noexpand\else
      \IfBooleanTF#1{\noexpand\@ctrerr}%
        {\noexpand\FN@orange##1}%
      \noexpand\fi
  }%
}
\def\FN@build@symboldefTM#1#2{%
  \def\@tempa{#1}%
  \ifx\@tempa\@tempb
  \else
    \toks@\expandafter{\the\toks@\or\TextOrMath{#1}{#2}}%
    \expandafter\FN@build@symboldefTM
  \fi
}
\def\FN@orange#1{%
  \ifFN@robust
    \@arabic#1%
    \@bsphack
    \PackageInfo{footmisc}{Footnote number \number#1 out of range}%
    \protect\@fnsymbol@orange
    \@esphack
  \else \@ctrerr \fi
}
\global\let\@diagnose@fnsymbol@orange\relax
\AtEndDocument{\@diagnose@fnsymbol@orange}
\def\@fnsymbol@orange{%
  \gdef\@diagnose@fnsymbol@orange{%
    \PackageWarningNoLine{footmisc}{Some footnote number(s)
      were out of range
      \MessageBreak
      see log for details%
    }%
  }%
}
\DefineFNsymbolsTM{bringhurst}{%
  \textasteriskcentered *%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textbardbl    \|%
  \textparagraph \mathparagraph
}%
\DefineFNsymbolsTM{chicago}{%
  \textasteriskcentered *%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textbardbl    \|%
```

```
  \#\#%
}%
\DefineFNsymbolsTM{wiley}{%
  \textasteriskcentered *%
  {\textasteriskcentered\textasteriskcentered}{**}%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textparagraph \mathparagraph
  \textbardbl    \|%
}%
\DefineFNsymbolsTM{lamport-robust}{%
  \textasteriskcentered *%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textparagraph \mathparagraph
  \textbardbl    \|%
  {\textasteriskcentered\textasteriskcentered}{**}%
  {\textdagger\textdagger}{\dagger\dagger}%
  {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}%
}
\DefineFNsymbolsTM*{lamport*}{%
  \textasteriskcentered *%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textparagraph \mathparagraph
  \textbardbl    \|%
  {\textasteriskcentered\textasteriskcentered}{**}%
  {\textdagger\textdagger}{\dagger\dagger}%
  {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}%
  {\textsection\textsection}{\mathsection\mathsection}%
  {\textparagraph\textparagraph}{\mathparagraph\mathparagraph}%
  {\textasteriskcentered\textasteriskcentered\textasteriskcentered}{***}%
  {\textdagger\textdagger\textdagger}{\dagger\dagger\dagger}%
  {\textdaggerdbl\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger\ddagger}%
  {\textsection\textsection\textsection}%%
    {\mathsection\mathsection\mathsection}%
  {\textparagraph\textparagraph\textparagraph}%%
    {\mathparagraph\mathparagraph\mathparagraph}%
}
\setfnsymbol{lamport*}
\DefineFNsymbolsTM{lamport*-robust}{%
  \textasteriskcentered *%
  \textdagger    \dagger
  \textdaggerdbl \ddagger
  \textsection   \mathsection
  \textparagraph \mathparagraph
  \textbardbl    \|%
  {\textasteriskcentered\textasteriskcentered}{**}%
  {\textdagger\textdagger}{\dagger\dagger}%
  {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}%
  {\textsection\textsection}{\mathsection\mathsection}%
```

```
  {\textparagraph\textparagraph}{\mathparagraph\mathparagraph}%
  {\textasteriskcentered\textasteriskcentered\textasteriskcentered}{***}%
  {\textdagger\textdagger\textdagger}{\dagger\dagger\dagger}%
  {\textdaggerdbl\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger\ddagger}%
  {\textsection\textsection\textsection}%%
    {\mathsection\mathsection\mathsection}%
  {\textparagraph\textparagraph\textparagraph}%%
    {\mathparagraph\mathparagraph\mathparagraph}%
}
\newcommand\mpfootnotemark{%
  \@ifnextchar[%]
    \@xmpfootnotemark
    {%
      \stepcounter\@mpfn
      \protected@xdef\@thefnmark{\thempfn}%
      \@footnotemark
    }%
}
\def\@xmpfootnotemark[#1]{%
  \begingroup
    \csname c@\@mpfn\endcsname #1\relax
    \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotemark
}

\endinput
⟨/footmisc⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

G

H

U

V

W

X