Reimplementation of LaTeX 2ε's heading commands using templates

LaTeX Project*

v0.9a 2026-01-14

Abstract

Contents

*Initial implementation by Frank Mittelbach.

1 Introduction

This module reimplements tagging aware heading commands with templates. The new implementation is used automatically if \DocumentMetadata is used and replaces patches and redefinitions done in the latex-lab-sec. In a later step both modules will be merged.

Most of the documentation is of interest only for class and package authors who want to setup their own heading commands but the new implementation changes also the user interface. This is documented first.

2 Changes to the user interface

2.1 New user commands

`\theheading` This command expands inside a heading to the current number.

`\partmark` This replaces the fix `\markboth{}{}` or similar in the standard `\part` definition.

2.2 The optional argument

The standard heading commands (re)defined by this module use the standard 2e syntax, e.g.,

```
\section[toc]{title}
```

The star-form `\section*{Title}` stops the numbering, toc, bookmark and running header as it was the way for the last 30-odd years. New is that the optional argument is handled as a key/val list if an equal sign at the outer level is detected, otherwise the argument is taken to be the value of the `shorttitle` and passed to the table of contents, the running headers, the bookmarks and used with `\nameref`. Note that this means, that

```
\section*[Title]{Title}
```

will create a toc entry and a bookmark!

The following keys can be used

`toc` This sets the text for the table of contents. It also forces that the entry is created. So

```
\section*[toc=Introduction]{Introduction to this document}
```

will create an unnumbered entry “Introduction” in the table of contents. An empty value (i.e., `toc=`) will suppress the toc entry.

`running` This sets the text for the running header and forces the use of the mark command, so even with a starred section the header will be updated. An empty value will suppress the call of the mark commands, so a header from a previous section will prevail.

`bookmark` This sets the text for the bookmarks (the PDF outline) if, e.g., hyperref is loaded. As with the previous keys and empty value suppresses the bookmark entry.

`nameref` This allows to set the text used for a crossreference with `\nameref`.

`label` This sets a label, so is an alternative to using a `\label` command.

`subtitle, quote` Theses keys will allow to pass a subtitle and a quote to the underlying code, but currently they are not yet used by the implementations of the standard LaTeX heading commands.

`shorttitle` There is the key which is used if the optional argument is not of key/val form. So,

```
\section[short]{long title}
```

is the same as

```
\section[shortitle=short]{long title}
```

The key sets the toc, running, bookmark and nameref text.

numbered, unnumbered This allow to control the numbering without stopping toc, running head or bookmarks as it would happen with the starred version of the heading command.

template keys Any other key present is assumed to be a key that should be passed to the heading instance to overwrite some of its settings. So, e.g.,

```
\section[placement=top,number-format=\fbox{\theheading}]{Title}
```

will force the section to start a new page and put the number into an `\fbox`. For a list of supported keys, check the following parts of the documentation.

2.3 Class support

The module redefines all heading commands of the three standard classes, article, report and book to use the new implementation.

Heading commands of other classes that are defined with `\@startsection` are supported (with some restrictions) through a compatibility layer described in the next section.

The heading commands `\part` and `\chapter` are typically defined with `\secdef` and the internal command `\@part` and `\@chapter`. The module redefines `\secdef` to use the standard instances for these commands—this adds tagging support but *changes the layout* until the class provides its own instances.

For the ams-classes, `latex-lab-firstaid` contains instances for `\part` and `\chapter`. They comes near to the original layout but not exactly.

Other heading commands defined with `\secdef` will error as the code has no real chance to know how to handle such a heading.

Heading commands which uses neither `\@startsection` or `\secdef` (e.g, the `\chapter` command of memoir, or the heading commands of KOMA classes) are not changed by this module and so will not be tagged correctly unless they add explicit tagging support.

2.4 Changing heading commands

It is possible to change heading commands by editing the instance they use. E.g., in the standard classes one could frame every heading number with

```
\EditInstance{heading}{section}
  {number-format=\fbox{\theheading}}
```

The name of the instance is the same as the heading command.

To support other classes which still setup their heading commands with `\@startsection` the code has a legacy compatibility layer: the `\@startsection` command has been redefined to setup instances on-the-fly at the first use of a heading command. These instances have names using the the first argument of `\@startsection` with an attached `-@startsection`. As they are created on-the-fly these internal instances can be only edited *after* the first use of the heading command. Also as the names of the instances are built from the first argument of `\@startsection` it is not possible to define two different

heading commands of the same level with `\@startsection` as that would require two instances with different names. The next section describes how to lift this restrictions by converting such sectioning commands to use the new interfaces.

2.5 Converting heading commands defined with `\@startsection`

To convert a heading command defined with `\@startsection` to the new interface suitable instances must be declared. The same needs to happen if one wants to alter the layout of a heading that was defined with `\@startsection` in the document preamble. How this can be done is demonstrated here with the `\section` command of the ltugboat class.

2.5.1 Step 1: Getting the template name and the instance values

The sectioning commands use two template *types*, `heading` and `headformat`. Compile the following document

```
\DocumentMetadata{}
\documentclass{ltugboat}

\begin{document}

\section{test} % <--- needed so that the instances get defined
\ShowInstanceValues{heading}{section-@startsection}
\ShowInstanceValues{headformat}{section-@startsection}

\end{document}
```

This will show the instance values in the log-file:

```
The instance 'section-@startsection' of type 'heading' has values:
>  level  =>  1
>  placement  =>  normal
>  mark-cmd  =>  \sectionmark {##1}
>  para-indent  =>  false
>  before-sep  =>  8.0pt plus 2.0pt minus 2.0pt
>  penalty  =>  \c_max_int
>  after-penalty-sep  =>  0pt
>  after-sep  =>  4.0pt
>  start-code  =>
>  final-code  =>
>  decls  =>  \tubsecfmt
>  number-decls  =>
>  title-decls  =>
>  subtitle-decls  =>
>  quote-decls  =>
>  headformat-instance  =>  section-@startsection
>  number-format  =>  \theheading
>  contents-extra  =>
>  name  =>  section
>  from template  =>  display.
```

```
The instance 'section-@startsection' of type 'headformat' has values:
>  indent  =>  0pt
>  before-code  =>  \tubsecfmt
>  after-code  =>
>  number-title-sep  =>  1em
>  from template  =>  hang.
```

2.5.2 Step 2: Get the default template values

The last line in both lists show after the `from template` the names of the templates of each type. That can be used to get the default values of these templates. Compile the following document:

```
\DocumentMetadata{}
\documentclass{ltugboat}

\begin{document}
\ShowTemplateDefaults{heading}{display}
\ShowTemplateDefaults{headformat}{hang}
\end{document}
```

This gives in the log and terminal:

```
The template 'display' of type 'heading' has default values:
>  level  =>  0
>  placement  =>  normal
>  mark-cmd  =>
>  para-indent  =>  false
>  before-sep  =>  0pt
>  penalty  =>  \c_max_int
>  after-penalty-sep  =>  0pt
>  after-sep  =>  0pt
>  start-code  =>
>  final-code  =>
>  decls  =>  \normalfont
>  number-decls  =>
>  title-decls  =>
>  subtitle-decls  =>
>  quote-decls  =>
>  headformat-instance  =>  hang
>  number-format  =>  \theheading
>  contents-extra  =>  .

The template 'hang' of type 'headformat' has default values:
>  indent  =>  0pt
>  before-code  =>
>  after-code  =>
>  number-title-sep  =>  1em.
```

2.6 Step 3: Declare the instances

By comparing the instance values with the default values one can see which values should be changed in the instance. The names of the new instances can be freely chosen best practice is to use the name of the heading command (without a backslash).

This leads then to these declarations:

```
\DeclareInstance{heading}{section}{display} %#1=heading, #2=name, #3=template name
 {
   level       =  1,
   mark-cmd    =  \sectionmark{#1}, % remove second hash
   before-sep  =  8.0pt plus 2.0pt minus 2.0pt,
   after-sep   =  4.0pt,
   decls       =  \tubsecfmt,
   headformat-instance = section, % new name
 }
\DeclareInstance{headformat}{section}{hang} %#1=heading, #2=name, #3=template name
 {
   before-code  =  \tubsecfmt
 }
```

2.6.1 Step 4 Redefine the `\section` command

At last the heading should be defined to use the new interface.

```
\DeclareDocumentCommand \section {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {section} {#1} {#2} {#3} }
```

3 Setting up LaTeX 2ε heading commands in classes

Heading commands are managed by defining an instance of the `heading` template and then using through `\ParseLaTeXeHeading`, e.g.,

```
  \DeclareDocumentCommand \part {s ={shorttitle}o m}
      { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
```

The name of the `heading` instance is given in the first argument of `\ParseLaTeXeHeading` and it is recommended that classes use the same name as the heading command, to make it easier for users to identify the instance to edit if they want to adapt the command. Examples of instances that mimic the behavior of the heading commands in the standard classes can be found below. Section **??** shows how to get instances from commands previously defined with `\@startsection`.

Legacy setup using `\@startsection` remains supported albeit not that performant and with some restrictions for the users, see section 2.4 above and in the documentation below.

In contrast `\secdef` is only rudimentarly supported. It delegates the layout to two commands which can contain arbitrary code (usually hardwired) so that there is no realistic chance to take this apart and figure out what values should be used for what template parameters. We therefore redefine `\secdef` and map the standard commands `\part` and `\chapter` to the standard instance and error if other uses are detected. Classes using `\secdef` should setup suitable instances that mimic their current layout, an example can be found for the ams-classes in `latex-lab-firstaid.dtx`.

4 Open points

- There is no good interface yet to change e.g. the font family of all heading commands in one go.
- Support for `titlesec`, see section **??**.
- Decide if `before-code` and `after-code` is combined into a format command.

5 Template types and templates for headings

The template(s) for headings expect(s) a large number of positional arguments containing document user data. The document-level command, e.g., `\section` may not offer all of them directly, but may do so via a key value interface or not at all. If no interface is provided then the template is passed `\NoNalue` . If it is offered via a key value interface, the template receives whatever is set by the user (and `\NoNalue` otherwise).

In my initial implementation I had only the main data as positional arguments, but I came to the conclusion that this scheme here is better going forward.

5.1 Template types

The template type `heading` has 10 data arguments, which is more than can be specified as positional arguments. For that reason argument #9 holds 2 brace groups. The alternative would be to specify such arguments as keys, but for a number of reasons I think the approach to put seldom necessary data arguments all in the last positional argument is actually better (besides being faster).

5.1.1 The template type 'heading'

Arg: 1 key/value list to alter the default heading parameters

Arg: 2 unnumbered heading?

Arg: 3 main title of the heading

Arg: 4 toc title

Arg: 5 running title

Arg: 6 bookmark title

Arg: 7 nameref text

Arg: 8 label for the heading in the form `\label{`⟨*string*⟩`}`

Arg: 9 { sub title } { quotation }

Semantics:

Handles the layout and processing aspects of a heading.

Whether or not the heading is numbered is governed through a boolean, expecting the result of an `s` specification of `\NewDocumentCommand` or equivalent, i.e., expects `\BooleanTrue` or `\BooleanFalse`.

If the title data is also used for bookmarks, toc, running header, and cross references then it has to be given several times which can be arranged for by the parsing interface command that calls the template instance.

If the bookmark, toc, or running argument is set to "empty" then the bookmark, toc or running header should be suppressed by the template. This enables the user on document level to explicitly specify, for example, `[bookmark=]` to suppress the bookmark.

The `nameref` argument is not expected to be empty. Its value should always be used as given if named references are to be generated.

The label argument either holds a `\label` command as provided by the user (or several, see implementation of `\ParseLaTeXeHeading`) or it is empty. I.e., it can be directly executed by a template at the right point where the reference counter (if any) has been set up without the need to check its content.

Argument #9 holds further user data (in brace groups) which are seldom implemented, but if they are the data is available in a positional argument, which then needs to be taken apart using `\@firstoftwo` (for subtitle) or `\@secondoftwo` (for a quotation). If no data is provided `\NoValue` is used to indicate that.

5.1.2 The template type 'headformat'

Arg: 1 key/value list to alter the default headformat parameters

Arg: 2 formatted heading number

Arg: 3 main title of the heading

Arg: 4 subtitle

Arg: 5 quotation

Semantics:

Handles the layout of just the label, main title, subtitle, and quotation but not the spatial relation to previous and following text.

Whether or not the heading is numbered is governed through a boolean, e.g., result of an `s` specification in `\NewDocumentCommand` or equivalent.

If there is no subtitle or quotation then this is indicated with `\NoValue`.

Note: instead of unnumbered + counter we could just have a single argument containing the number representation (or `\NoValue` if not used). The reason that there are two is that this allows us to continue to support `\@seccntformat`, but I'm not sure this is a good enough reason.

5.2 Templates

There are a number of keys that are expected to be recognized by all heading templates (though they may choose not to make use of them). These are listed below instead of being repeated on the actual templates.

All other keys are either attached to the `heading` or to the `headformat` templates. Those that typically vary from heading instance to the next (e.g., `decls`) are all declared in the `heading` templates even if they are actually only used within `headformat` templates. In other words, `headformat` templates have a number of implicit variables that they expect to be set.[1]

5.2.1 The `heading` template '⟨*all*⟩'

Attributes:

`name` (*tokenlist*) Referencable name of the heading instance. String that is acceptable in csnames for use in building counter names, etc.

`parent-name` (*tokenlist*) Name of the next higher heading instance. If not given, then the internal heading level of the heading instance is set to `0`

`reset-counter` (*tokenlist*) Name of the heading instance that should reset the numbering of this heading level (if any)

`level` (*integer*) Sets the internal heading-level rather than deducing it from `parent-name`. Can be used to specify the top-level heading if not `0`, or all headings in legacy implementations, e.g., through `\@startsection`

`placement` (*choice*) Set the heading placement, i.e., the behavior of the heading with respect to page breaks. Allowed values are `page` (heading forms a page if its own), `top` (heading starts a new page), `normal` (heading can appear anywhere on the page). Further possibilities might be `rectopage` and `rectotop` if we implement that. Default: `normal`

`start-code` (*tokenlist*) Default value is set by the `placement` key. Executed before the heading starts, so can issue, for example, a `\clearpage`

`final-code` (*tokenlist*) Default value is set by the `placement` key. Executed after the heading is typeset. Can set up code for putting the heading on a page by its own, or arrange for paragraph handling of a following paragraph, etc.

`mark-cmd` (*function(1)*) Function that receives the ⟨*running*⟩ argument and creates a suitable mark insertion Default: `\`⟨*name*⟩`mark`

Semantics & Comments: The above keys should be implemented by all heading templates.

At the moment I have retained the LaTeX 2ε interfaces for marks, e.g., one has to set up `\chaptermark`, `\sectionmark`, etc. but I'm not sure this should stay (even though it is is certainly simpler from a compatibility perspective.

All templates should set up `\theheading` to correspond to `\the`⟨*name*⟩.

[1] Maybe questionable

5.2.2 The `heading` template 'display'

Attributes:

`para-indent` (*boolean*) Should the paragraph after the heading be indented? Default: `false`

`before-sep` (*skip*) Vertical space before the heading if there is no page or column break. If there is one it vanishes. In particular this means it will not be used in the heading `placements` `page` or `top` Default: `0pt`

`penalty` (*integer*) Penalty to break before the heading. The default (TEX's largest integer) indicates that no penalty was set in which case `\@secpenalty` is used. Default: `1073741823`

`after-penalty-sep` (*skip*) Vertical space before the heading but after the penalty for the heading. If the penalty results in a page or column break, this space remains at the top of the page Default: `0pt`

`after-sep` (*skip*) Vertical space after the heading Default: `0pt`

`decls` (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Default: `\normalfont`

`number-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading number, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`title-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading title, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`subtitle-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading subtitle, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`quote-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading quote, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`number-format` (*function(1)*) Code that produces a formatted version of the heading number including any ornamentations. Its argument is the counter name for the head. However, instead of using a specific counter representations, e.g., `\thesection`, it can refer to the counter representation for the current heading counter via `\theheading` and ignore the argument. Default: `\theheading`

`contents-extra` (*tokenlist*) Code containing `\addcontents` calls to write to files like `.lot` or `.lot` Default: ⟨*empty*⟩

`headformat-instance` (*instance*) Template instances of type `headformat` Default: `hang`

Semantics & Comments: Several of the key names are simply bad and need revision!

5.2.3 The `heading` template 'runin'

Attributes:

`before-sep` (*skip*) Vertical space before the heading if there is no page or column break. If there is one it vanishes. Default: `0pt`

`penalty` (*integer*) Penalty to break before the heading. The default (TEX's largest integer) indicates that no penalty was set in which case `\@secpenalty` is used. Default: `1073741823`

`after-penalty-sep` (*skip*) Vertical space before the heading but after the penalty for the heading. If the penalty results in a page or column break, this space remains at the top of the page. It is also applied if the heading is a `page` or `top` heading. Default: `0pt`

`after-sep` (*skip*) Vertical space after the heading Default: `0pt`

`decls` (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Default: `\normalfont`

`number-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading number, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`title-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading title, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`subtitle-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading subtitle, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`quote-decls` (*tokenlist*) Declarations (such as font or color settings) applied to the heading quote, overwriting the setting of `decls`. Default: ⟨*empty*⟩

`number-format` (*function(1)*) Code that produces a formatted version of the heading number including any ornamentations. Its argument is the counter name for the head. However, instead of using a specific counter representations, e.g., `\thesection`, it can refer to the counter representation for the current heading counter via `\theheading` and ignore the argument. Default: `\theheading`

`headformat-instance` (*instance*) Template instances of type `headformat` Default: `hang`

Semantics & Comments: Several of the key names are simply bad and need revision!

5.2.4 The `headformat` template 'hang'

Attributes:

`indent` (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: `0pt`

`before-code` (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. The code can take an argument, e.g., `\MakeUppercase` Default: ⟨*empty*⟩

`after-code` (*tokenlist*) Code executed directly at the end of the main title text
Default: ⟨*empty*⟩

`number-title-sep` (*tokenlist*) Token list that can be used in the assignment to a TEX dimension (can contain `em` or `ex` values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `1em`

Semantics & Comments: Implements a heading layout where number and title start on the same line and the title is hanging off from that if the title has more than one line. It is what LATEX always used by default for `\section`, `\subsection`, and `\subsubsection`.
Several of the key names are simply bad and need a revision!

5.2.5 The `headformat` template 'runin'

Attributes:

`indent` (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: `0pt`

`before-code` (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. Default: ⟨*empty*⟩

`after-code` (*tokenlist*) Code executed directly at the end of the main title text
Default: ⟨*empty*⟩

`number-title-sep` (*tokenlist*) Token list that can be used in the assignment to a TEX dimension (can contain `em` or `ex` values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font.

In the hang template it is a horizontal dimension! Default: `1em`

Semantics & Comments: Implements a heading layout where number and title start on the same line but then continue with the following paragraph on the same line (or the next if the title overflows, i.e., the title is run-in. It is what LATEX always used by default for `\paragraph` and `\subparagraph`.
Several of the key names are simply bad and need a revision!

5.2.6 The `headformat` template 'display'

Attributes:

`indent` (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: `0pt`

`before-code` (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. Default: ⟨*empty*⟩

`after-code` (*tokenlist*) Code executed directly at the end of the main title text
Default: ⟨*empty*⟩

`number-title-sep` (*tokenlist*) Token list that can be used in the assignment to a TEX dimension (can contain `em` or `ex` values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font.

In the display template it is a vertical dimension! Default: `20 pt`

Semantics & Comments: Implements a heading layout where number and title are vertically separated. It is what LATEX always used by default for `\chapter` and `\part`.

Several of the key names are simply bad and need a revision!

5.3 Default instances

These instances are set up to mimic the design of the standard LATEX classes. For other classes they could be adjusted by changing the instance values and/or by basing them on a different template.

5.3.1 Instances of `heading` templates

`part` (instance of `display` template) Used for the `\part` command.

`chapter` (instance of `display` template) Used for the `\chapter` command.

`section` (instance of `display` template) Used for the `\section` command.

`subsection` (instance of `display` template) Used for the `\subsection` command.

`subsubsection` (instance of `display` template) Used for the `\subsubsection` command.

`paragraph` (instance of `runin` template) Used for the `\paragraph` command.

`subparagraph` (instance of `runin` template) Used for the `\subparagraph` command.

5.3.2 Instances of `headformat` templates

`display` (instance of `display` template) Used in `heading` instance for `\part` and `\chapter`.

`hang` (instance of `hang` template) Used in `heading` instance for `\section`, `\subsection`, and `\subsubsection`.

`paragraph` (instance of `runin` template) Used in `heading` instance for `\paragraph`

`subparagraph` (instance of `runin` template) Used in `heading` instance for `\subparagraph`

6 Support for legacy classes and packages

6.1 Support classes based on legacy LaTeX 2ε interfaces

\@startsection The \@startsection command has the following syntax:

```
\@startsection{name}{level}{indent}{beforeskip}{afterskip}{style}
```

with a special logic that the sign of ⟨*beforeskip*⟩ determines display or runin heading and that of ⟨*afterskip*⟩ whether or not the next paragraph is indented.

All arguments can be cleanly mapped to `heading` and `headformat` templates. Thus, if encountered suitable instances are declared unless already existing.

\secdef In contrast \secdef only does the parsing and delegates the layout to two commands which can contain arbitrary code (usually hardwired) so that there is no realistic chance to take this apart and figure out what values should be used for what template parameters.

We therefore do not attempt to model this, but instead just use the standard layout from LaTeX's standard classes for \chapter and \part if we can identify that the \secdef is used to define one of these.

Maybe we can try at some stage to get some static analysis tool going.

6.2 Support for the titlesec package interfaces

TODO

\titleformat The \titleformat declaration has the following syntax:

```
\titleformat{cmd}[shape]{format}{label}{sep}{before-code}[after-code]
```

\titlespacing The \titlespacing declaration has the following syntax:

```
\titlespacing*{cmd}{left-sep}{before-sep}{after-sep}[right-sep]
```

7 Support for links

All heading commands (numbered and unnumbered) should contain support for active links directly. hyperref does not patch the new heading commands! As \refstepcounter is not used there is not automatic target.

This means that all definitions should contain at an appropriate place a \MakeLinkTarget command. Typically numbered heading commands will use \MakeLinkTarget{⟨*counter*⟩} while unnumbered heading commands use \MakeLinkTarget[⟨*counter*⟩]{}.

The definition must take care that the target is not separated from the heading by a page break. It also should not affect spacing. The target should be placed so that a jump to the target gives a satisfying user experience, in most cases the best places is at the left margin a bit above the heading.

The targets create also structure destinations that are used by the tagging code. It is therefore important that the targets are in the right place in relation to the tagging commands.

8 Bookmarks

Bookmarks are created by the `\addcontentsline` command, more precisely in the new latex-lab toc code a hook with arguments is inserted at the begin of the command which contains the command that creates the bookmark. The arguments of `\addcontentsline` and so also of the hook are the type of the toc, the level name and the (short-)title of the heading. To pass a differing bookmark text the code uses `\texorpdfstring` in the `\addcontentsline`.

9 Tagging support

Tagging of heading commands has to do two tasks:

- surround the whole section (heading and text) with a `Sect` structure
- tag the heading with an Hn structure.

This is done with tagging sockets which are documented in the code and in `latex-lab-sec`.

10 Debugging support

`\DebugHeadingsOn`
`\DebugHeadingsOff`
`\head_debug_on:`
`\head_debug_off:`

These commands enable/disable debugging messages.

11 The Implementation

```
1 ⟨*package⟩
2 ⟨@@=head⟩
3 \ProvidesExplPackage
4  {latex-lab-testphase-sec-template}
5  {\ltlabsecIIdate}
6  {\ltlabsecIIversion}
7  {heading implementation}
```

11.1 Debugging

`\g__head_debug_bool`

```
8 \bool_new:N \g__head_debug_bool
```

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

`\__head_debug:n`
`\__head_debug_typeout:n`

```
9 \cs_new_eq:NN \__head_debug:n \use_none:n
10 \cs_new_eq:NN \__head_debug_typeout:n \use_none:n
```

(*End of definition for* `\__head_debug:n` *and* `\__head_debug_typeout:n`.)

`\head_debug_on:`
`\head_debug_off:`
`\__head_debug_gset:`

```
\cs_new_protected:Npn \head_debug_on:
  {
    \bool_gset_true:N \g__head_debug_bool
    \__head_debug_gset:
  }
\cs_new_protected:Npn \head_debug_off:
  {
    \bool_gset_false:N \g__head_debug_bool
    \__head_debug_gset:
  }
\cs_new_protected:Npn \__head_debug_gset:
  {
    \cs_gset_protected:Npx \__head_debug:n ##1
      { \bool_if:NT \g__head_debug_bool {##1} }
    \cs_gset_protected:Npx \__head_debug_typeout:n ##1
      { \bool_if:NT \g__head_debug_bool { \typeout{[head]~ ##1} } }
  }
```

(*End of definition for* `\head_debug_on:`, `\head_debug_off:`, *and* `\__head_debug_gset:`. *These functions are documented on page 16.*)

`\DebugHeadingsOn`
`\DebugHeadingsOff`

```
\cs_new_protected:Npn \DebugHeadingsOn  { \head_debug_on:  }
\cs_new_protected:Npn \DebugHeadingsOff { \head_debug_off: }
\DebugHeadingsOff
```

(*End of definition for* `\DebugHeadingsOn` *and* `\DebugHeadingsOff`. *These functions are documented on page 16.*)

`\__head_show_arguments:nnnnnn` Some debug data ...

```
\cs_new:Npn \__head_show_arguments:nnnnn #1#2#3#4#5 {
  \__head_debug_typeout:n{---------- Headformat~ instance~ arguments:}
  \__head_debug_typeout:n{1:~ keys~ =~ \exp_not:n{#1}}
  \__head_debug_typeout:n{2:~ number~ =~ \exp_not:n{#2}}
  \__head_debug_typeout:n{3:~ title~ =~ \exp_not:n{#3}}
  \__head_debug_typeout:n{4:~ subtitle~ =~ \exp_not:n{#4}}
  \__head_debug_typeout:n{5:~ quotation~ =~ \exp_not:n{#5}}
}
```

(*End of definition for* `\__head_show_arguments:nnnnnn`.)

`\__head_show_arguments:nnnnnnnnn` Some debug data ...

```
\cs_new:Npn \__head_show_arguments:nnnnnnnnn #1#2#3#4#5#6#7#8#9 {
  \__head_debug_typeout:n{---------- Heading~ instance~ arguments:}
  \__head_debug_typeout:n{1:~ keys~ =~ \exp_not:n{#1}}
  \__head_debug_typeout:n{2:~ unnumbered~ =~ \exp_not:n{#2}}
  \__head_debug_typeout:n{3:~ title~ =~ \exp_not:n{#3}}
  \__head_debug_typeout:n{4:~ toc~ =~ \exp_not:n{#4}}
  \__head_debug_typeout:n{5:~ running~ =~ \exp_not:n{#5}}
  \__head_debug_typeout:n{6:~ bookmark~ =~ \exp_not:n{#6}}
  \__head_debug_typeout:n{7:~ nameref~ =~ \exp_not:n{#7}}
  \__head_debug_typeout:n{8:~ label~ =~ \exp_not:n{#8}}
```

```
  \__head_debug_typeout:n{9:~ subtitle | quote ~ =~ \exp_not:n{#9}}
}
```

(*End of definition for* `\__head_show_arguments:nnnnnnnnn`.)

11.2 Temp variable(s)

`\l__head_tmpa_tl`

```
\tl_new:N\l__head_tmpa_tl
```

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

11.3 variable(s)

These token lists are automatically declared by the key machinery.

```
%\tl_new:N \l__head_bookmark_tl
%\tl_new:N \l__head_running_tl
%\tl_new:N \l__head_toc_tl
%\tl_new:N \l__head_nameref_tl
%\tl_new:N \l__head_subtitle_tl
%\tl_new:N \l__head_quote_tl
%\bool_new:N l__head_unnumbered_bool
```

`\l__head_label_tl`
`\l__head_instance_keys_tl`
`\l__head_typeset_number_tl`
`\l__head_saved_secnumdepth_tl`
`\l__head_title_tl`
`\l__head_placement_tl`

But this token lists needs a declaration:

```
\tl_new:N\l__head_label_tl
\tl_new:N\l__head_instance_keys_tl
\tl_new:N\l__head_typeset_number_tl
\tl_new:N\l__head_saved_secnumdepth_tl
\tl_new:N \l__head_title_tl
\tl_new:N \l__head_placement_tl
```

(*End of definition for* `\l__head_label_tl` *and others.*)

11.4 New user commands

`\theheading` This command expands to `\thechapter`, `\thesection` etc in every heading command.

```
\tl_new:N\theheading
```

(*End of definition for* `\theheading`. *This function is documented on page* **??**.)

`\partmark` This replaces the fix `\markboth{}{}` or similar in the standard `\part` definition. As the command is already defined in some classes (like memoir), we provide it through a hook.

```
\AddToHook{class/after}{\providecommand*\partmark[1]{\markboth{}{}}}
```

(*End of definition for* `\partmark`. *This function is documented on page* **??**.)

11.5 The LaTeX 2ε parsing interface

LaTeX 2ε provides heading commands with a starred form (unnumbered) and one optional argument (to specify an alternative toc/running head). We augment this slightly by supporting a key value interface in the optional argument. This is handled by defining the commands through `\ParseLaTeXeHeading`. This should happen in the document class.

`\ParseLaTeXeHeading`

`\ParseLaTeXeHeading` arguments:

1: instance name to use (of type "heading")

2: numbered? boolean

3: shorttitle or key/val for layout adjustments and individual title settings

4: main title

This command handles the document input that may be hidden in the optional argument, i.e., special data for toc, running (header), bookmark, label, and for more specialized headings subtitle and quote. It also handles the legacy case that the optional argument is just an alternate title to be used for toc, running, and bookmark (through the key shorttitle to which it gets converted).

```
\cs_new_protected:Npn \ParseLaTeXeHeading #1 #2 #3 #4 {
  \__head_debug_typeout:n{==================================}
  \__head_debug_typeout:n{#1:~ \IfBooleanT{#2}{*}
                \IfValueT{#3}{[\exp_not:n{#3}]}
                {\exp_not:n{#4}}}
```

We first check if the title argument contains a `\label` command. If yes, we remove it and add it to `\l__head_label_tl` (and if more than one all of them) and the rest of the main title in `\l__head_title_tl` for later use. If there was no `\label` command then `\l__head_title_tl` is set to `#4`.

```
  \__head_find_label:w #4 \label\q_no_value \__head_find_label:w
```

By default toc, running, and bookmark use the data provided by the main title. However, if the second argument is true (i.e., a star was given) they are all suppressed (because this is the LaTeX 2ε logic).

```
  \IfBooleanTF{#2}
    {
      \tl_clear:N \l__head_toc_tl
      \tl_clear:N \l__head_running_tl
      \tl_clear:N \l__head_bookmark_tl
      \bool_set_true:N \l__head_unnumbered_bool
    }
    {
      \tl_set_eq:NN \l__head_toc_tl      \l__head_title_tl
      \tl_set_eq:NN \l__head_running_tl  \l__head_title_tl
      \tl_set_eq:NN \l__head_bookmark_tl \l__head_title_tl
      \bool_set_false:N \l__head_unnumbered_bool
    }
```

The nameref always starts out matching the main title.

```
  \tl_set_eq:NN \l__head_nameref_tl \l__head_title_tl
```

Normally we don't have a subtitle or quote unless they are explicitly given through keys, so we start with `\c_novalue_tl`

```
\tl_set_eq:NN \l__head_subtitle_tl \c_novalue_tl
\tl_set_eq:NN \l__head_quote_tl    \c_novalue_tl
```

If the optional argument is empty we set the `\l__head_instance_keys_tl` to empty, otherwise process the key/val list setting the keys defined in the module "head". This overwrites the defaults specified above if keys like "toc" are in the list. All the key/vals unknown by that module are put into `\l__head_instance_keys_tl` which may or may not make this token list non-empty.
If the user has a `label` key and also a `\label` in the main title argument then the latter is ignored (no error checking for that). We could alternatively set all labels we find, perhaps that is the better approach?

```
\IfNoValueTF { #3 }
  { \tl_clear:N \l__head_instance_keys_tl }
  { \keys_set_known:nnN {heading} { #3 } \l__head_instance_keys_tl }
```

Finally we call the heading instance using the collected mandatory arguments. To simplify downstream processing we pass the values not the container tokenlists resulting from the above processing.[2]

```
\__head_debug_typeout:n{use~ 'heading'~ instance:~ #1 }
\use:e {
  \exp_not:N \UseInstance{heading}{#1}
             { \exp_not:o \l__head_instance_keys_tl }
             { \bool_if:NTF \l__head_unnumbered_bool \BooleanTrue \BooleanFalse }
             { \exp_not:o \l__head_title_tl}
             { \exp_not:o \l__head_toc_tl }
             { \exp_not:o \l__head_running_tl }
             { \exp_not:o \l__head_bookmark_tl }
             { \exp_not:o \l__head_nameref_tl }
             { \exp_not:o \l__head_label_tl }
             { { \exp_not:o \l__head_subtitle_tl }
               { \exp_not:o \l__head_quote_tl    } }
}
}
```

Syntax keys that can appear in the optional argument of headings parsed by `\ParseLaTeXeHeading`. All other keys used there are passed to the heading instance to overwrite instance setting.

```
\keys_define:nn {heading} {
  , shorttitle .meta:n    =
      {toc = {#1} , running = {#1} , bookmark = {#1} , nameref= {#1} }
  , bookmark   .tl_set:N  = \l__head_bookmark_tl
  , running    .tl_set:N  = \l__head_running_tl
  , toc        .tl_set:N  = \l__head_toc_tl
  , nameref    .tl_set:N  = \l__head_nameref_tl
%
  , numbered   .bool_set_inverse:N = \l__head_unnumbered_bool
  , numbered   .default:n = true
  , unnumbered .bool_set:N = \l__head_unnumbered_bool
  , unnumbered .default:n = true
%
```

[2] I think this is a common requirement and perhaps `\UseInstance` should really o-expand all arguments it receives.

```
    , subtitle   .tl_set:N  = \l__head_subtitle_tl
    , quote      .tl_set:N  = \l__head_quote_tl
```

We collect labels together with their `\label` command in case there is more than one. This way we can later simply execute `\l__head_label_tl` without any status checks.

```
    , label      .code:n    = \tl_put_right:Nn \l__head_label_tl { \label{#1} }
}
```

(*End of definition for* `\ParseLaTeXeHeading`. *This function is documented on page* ??.)

`\__head_find_label:w` A simple-minded check for an existing `\label` command in the main title (could be done better I guess). We add `\label\q_no_value` at the end of the argument, so that we can be sure to find something.

As currently implemented the spaces on both sides of a `\label` command survive if it is removed. This may be an issue in which case this may need some adjustments.

```
\cs_new:Npn \__head_find_label:w #1 \label #2#3 \__head_find_label:w {
```

If the token after `\label` is `\q_no_value` then the title had no label and we set the two variables accordingly.

```
  \quark_if_no_value:nTF {#2}
     {
       \__head_debug_typeout:n{----~no~label }
       \tl_clear:N \l__head_label_tl
       \tl_set:Nn\l__head_title_tl {#1#3}
     }
```

Otherwise, there was a real `\label` command and `#2` holds the label string.

```
     {
       \__head_debug_typeout:n{----~label~found~#2}
       \tl_set:Nn\l__head_label_tl { \label{#2} }
```

To construct the value for `\l__head_title_tl` we have to get rid of the two tokens we added at its end, this is done with `\__head_find_label_aux:w`.

```
       \tl_set:Nn\l__head_title_tl {#1}
       \__head_find_label_aux:w #3\__head_find_label_aux:w
     }
     \__head_debug_typeout:n{----~return:~ '\exp_not:o \l__head_title_tl' }
}
```

There is at least one more `\label` now and the token after it should be our `\q_no_value`. If not then the title argument had at least 2 labels and we collect the newly found one and recurse.

```
\cs_new:Npn \__head_find_label_aux:w #1 \label #2#3 \__head_find_label_aux:w {
```

The `#1` may not be everything we have to pick up but we know for sure that it belongs to the title.

```
  \tl_put_right:Nn \l__head_title_tl { #1 }
```

Now let's see if we are at the end of the argument. If not pick up the newly found `\label` and recurse on the remaining material.

```
  \quark_if_no_value:nF {#2}
    {
      \__head_debug_typeout:n{----~ extra~ label~ '#2'~ found }
      \tl_put_right:Nn \l__head_label_tl { \label{#2} }
      \__head_find_label_aux:w #3\__head_find_label_aux:w
    }
}
```

(*End of definition for* `\__head_find_label:w`.)

11.6 Templates

11.6.1 Template types

`heading` (*type*) Templates of type `heading` are used to produce headings. The positional arguments are:

```
1: key/val list
2: unnumbered?
3: title
4: toc
5: running
6: bookmark
7: nameref
8: label(s)
9: { subtitle } { quote }
```

```
\NewTemplateType{heading}{9}
```

`headformat` (*type*) Templates of type `headformat` are used to produce heading layout from the formatted heading number (if any), the heading title, subtitle and quotation. The positional arguments are:

```
1: key/val list for document-level customizations
2: formatted heading number
3: title
4: subtitle
5: quote
```

```
\NewTemplateType{headformat}{5}
```

11.6.2 heading templates interfaces

We have two heading templates: display and runin.

`heading display` (*templ.*) The `display` template produces a display heading, i.e., one that has vertical space before and after it.

```
\DeclareTemplateInterface{heading}{display}{9}
{
  , name          : tokenlist
  , parent-name   : tokenlist
  , reset-counter : tokenlist
  , level         : integer = 0
```

The `placement` key sets default values for the keys `start-code` and `final-code`.

```
  , placement         : choice {page , top , normal } = normal
  , mark-cmd      : function(1) =
  %
  % many more keys for layout settings are missing for now
  , para-indent  : choice { true , false } = false
```

We use `\c_max_int` as a fake penalty to indicate that no penalty was given in a key.

```
  , before-sep    : skip = 0pt
  , penalty       : integer = \c_max_int
  , after-penalty-sep : skip = 0pt
  , after-sep     : skip = 0pt
```

The next two keys are only there to overwrite the `placement` settings with explicit code. Thus, their default value is set up by the `placement` key (which has a default).

```
  , start-code    : tokenlist =      % no default values!
  , final-code    : tokenlist =      % no default values!

  , decls          : tokenlist = \normalfont
  , number-decls   : tokenlist =
  , title-decls    : tokenlist =
  , subtitle-decls : tokenlist =
  , quote-decls    : tokenlist =

  , headformat-instance : tokenlist = hang
  , number-format  : function(1) = \theheading
  , contents-extra : tokenlist =
}
```

`heading runin` (*templ.*) This template is similar to the `display` template but implements a runin heading, i.e., one where the paragraph text continues on the same line.
Unfortunately, we can't really use `\DeclareTemplateCopy` to set it up because with `\EditTemplateDefaults` we are not able to alter the setup for the `placement` and `para-indent` keys as that involves changing the implementation code. Thus with `\DeclareTemplateCopy` we can copy the interface setup but the code setup still needs to be done using `\DeclareTemplateCode`.

```
\DeclareTemplateCopy{heading}{runin}{display}
```

11.6.3 headformat templates interfaces

We have three template: display, hang and runin.

`headformat display` (*templ.*) The `display` template produces

```
\DeclareTemplateInterface{headformat}{display}{5}
{
  , indent           : length = 0pt
  , before-code      : tokenlist =
  , after-code       : tokenlist =
  , number-title-sep : tokenlist = 20pt
}
```

`headformat hang` (*templ.*) The `hang` template produces

```
\DeclareTemplateInterface{headformat}{hang}{5}
{
  , indent           : length = 0pt
  , before-code      : tokenlist =
  , after-code       : tokenlist =
  , number-title-sep : tokenlist = 1em
}
```

`headformat runin` (*templ.*) The `runin` template produces

```
\DeclareTemplateInterface{headformat}{runin}{5}
{
  , indent           : length = 0pt
  , before-code      : tokenlist =
  , after-code       : tokenlist =
  , number-title-sep : tokenlist = 1em
}
```

11.6.4 heading templates code

heading display (*templ.*)

```
\DeclareTemplateCode{heading}{display}{9}
{
  , name          = \l__head_name_tl
  , level         = \l__head_level_int
% next two not yet used
  , parent-name   = \l__head_pname_tl
  , reset-counter = \l__head_reset_cnt_tl
```

The `placement` key determines whether the heading can appear anywhere (`normal`), automatically starts a new page or column (`top`), or is on a page of its own (`page`.
It is implemented by setting `\l__head_start_code_tl` and `\l__head_final_code_tl`. These settings can be fine-tuned or overwritten with the keys `start-code` and `final-code`, if necessary.

```
  , placement         = {
        page   = \__head_debug_typeout:n{ A~ page~ heading }
                 \tl_set:Nn \l__head_placement_tl { page }
       ,top    = \__head_debug_typeout:n{ A~ top~ heading }
                 \tl_set:Nn \l__head_placement_tl { top }
       ,normal = \__head_debug_typeout:n{ A~ normal~ heading }
                 \tl_set:Nn \l__head_placement_tl { normal }
                   }
  , mark-cmd      = \__head_mark_cmd:n
```

For now we make use of the legacy coding for indentation of the following paragraph.

```
  , para-indent  = {
                      true  = \@afterindenttrue
                     ,false = \@afterindentfalse
                   }
  , before-sep    = \l__head_before_skip
  , penalty       = \l__head_penalty_int
  , after-penalty-sep  = \l__head_after_penalty_skip
  , after-sep     = \l__head_after_skip
  , start-code    = \l__head_start_code_tl
  , final-code    = \l__head_final_code_tl
  , headformat-instance = \l__head_headformat_instance_tl
  , decls          = \l__head_decls_tl
  , number-decls   = \l__head_number_decls_tl
  , title-decls    = \l__head_title_decls_tl
  , subtitle-decls = \l__head_subtitle_decls_tl
  , quote-decls    = \l__head_quote_decls_tl
  , number-format  = \__head_number_format:n
  , contents-extra = \l__head_contents_extra_tl
}
{
  \tl_set_eq:Nc \theheading { the \l__head_name_tl }
```

We store the value of `secnumdepth` so that we can change it locally.

```
  \tl_set:Ne\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}
  \__head_show_arguments:nnnnnnnnn
       {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
```

It would be nice if there is a variant of `\SetTemplateKey` in which the template type and name are implicit, e.g., `\SetCurrentTemplateKeys` because this is usually what I need.

First evaluate any key setting done by the user in the optional first argument.

```
\tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{display}{#1} }
```

Based on the `placement` key we set up `\l__head_start_code_tl` and `\l__head_final_code_tl`. These two variables can also be set with the keys `start-code` and `final-code` in which case we don't alter the definition.

```
\tl_if_empty:oT \l__head_start_code_tl
 { \str_case:VnF \l__head_placement_tl
    {
      { page }
      { \tl_set:Nn \l__head_start_code_tl
```

Clearly not `\@tempswa` and the page styles should be adjustable.

Straight from the placement definition of `\part`.

```
          {
           \if@openright
             \cleardoublepage
           \else
             \clearpage
           \fi
           \thispagestyle{plain}%
           \if@twocolumn
             \onecolumn
             \@tempswatrue
           \else
             \@tempswafalse
           \fi
           \null\vfil
          }
      }
      { top }
      { \tl_set:Nn \l__head_start_code_tl
          {
            \if@openright\cleardoublepage\else\clearpage\fi
            \thispagestyle{plain}%
            \global\@topnum\z@
          }
      }
    }
    { \tl_clear:N \l__head_start_code_tl }
 }
%
 \tl_if_empty:oT \l__head_final_code_tl
  { \str_case:VnF \l__head_placement_tl
     {
       { page }
       { \tl_set:Nn \l__head_final_code_tl
           {
             \vfil\newpage
             \if@twoside
               \if@openright
                 \null
                 \thispagestyle{empty}%
               \newpage
               \fi
```

```
              \fi
              \if@tempswa
                \twocolumn
              \fi
            }
        }
      }
      { \tl_set:Nn  \l__head_final_code_tl { \@afterheading } }
  }
```

Then we set up the penalty to use (might be given as a key value).

```
  \__head_determine_penalty:
  \__head_determine_number_typesetting:N #2
```

Next comes the vertical spacing and penalty before the heading. This includes running `\l__head_start_code_tl` if it contains any code, e.g., to start a new page.

```
  \__head_vertical_before_spacing:
```

We are in vmode now and here is the point where we (with tagging) can close a previous Sect structure and open the new one.

```
\UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
\UseTaggingSocket{sec/begin}
  {{\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
```

Up to this point everything is identical for both display and runin headings. But from now on they have their own code. We have dealt with argument #1 and #2 so now we can unbundle argument #9 so that it is easier to process downstream

```
  \__head_debug_typeout:n{use~ 'headformat'~instance:~
                          \l__head_headformat_instance_tl }
  \use:e {
    \UseInstance{headformat} { \l__head_headformat_instance_tl }
                 { \exp_not:o \UnusedTemplateKeys }
                 { \exp_not:o { \l__head_typeset_number_tl } }
                 { \exp_not:n { #3 } }
                 { \exp_not:o { \use_i:nn  #9 } }
                 { \exp_not:o { \use_ii:nn  #9 } }
  }
 %
% --- handle marks, toc-entry, bookmark, nameref, and label
%
  \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
%
% --- post-heading handling (vertical)
```

Restore `secnumdepth`

```
  \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
  \par \nobreak
  \skip_vertical:N \l__head_after_skip
% --- prepare next paragraph (defaults to \cs{@afterheading}
%
  \l__head_final_code_tl
%
  \ignorespaces
}
```

```
\AddToHook{begindocument}{
  \ifcsname if@openright\endcsname
  \else
```

Need to hide this a little if it is in fact already defined! UFI: why not simply `\newif`??

```
    \expandafter\newif\csname if@openright\endcsname
  \fi
}
```

(*End of definition for .*)

`heading runin` (*templ.*) The `runin` template is very similar to the `display` one.

```
\DeclareTemplateCode{heading}{runin}{9}
 {
  , name          = \l__head_name_tl
  , level         = \l__head_level_int
% next two not yet used
  , parent-name   = \l__head_pname_tl
  , reset-counter = \l__head_reset_cnt_tl
```

It wouldn't make sense to have page placement with a runin heading since there would be nothing to run into.

```
  , placement         = {
        page   = \typeout{ ^^JA~ runin~ page~ placement~ heading~makes~no~sense
                           (top~used)}
                 \tl_set:Nn \l__head_placement_tl { top }
       ,top    = \__head_debug_typeout:n{ A~ top~ heading }
                 \tl_set:Nn \l__head_placement_tl { top }
       ,normal = \__head_debug_typeout:n{ A~ normal~ heading }
                 \tl_set:Nn \l__head_placement_tl { normal }
                 }
  , mark-cmd      = \__head_mark_cmd:n
```

In a runin heading you can't set up paragraph indentation of the following paragraph (since that one is "run in". But we accept the key and just spit out a warning.

UFi:this types out messages for all run-in headers! Therefore disabled for now

```
  , para-indent  =  {
                      true  = %\typeout{para-indent~ setting~ ignored}
                     ,false = %\typeout{para-indent~ setting~ ignored}
                    }
  , before-sep    = \l__head_before_skip
  , penalty       = \l__head_penalty_int
  , after-penalty-sep  = \l__head_after_penalty_skip
  , after-sep     = \l__head_after_skip
  , start-code    = \l__head_start_code_tl
  , final-code    = \l__head_final_code_tl
  , headformat-instance = \l__head_headformat_instance_tl
  , decls          = \l__head_decls_tl
  , number-decls   = \l__head_number_decls_tl
  , title-decls    = \l__head_title_decls_tl
  , subtitle-decls = \l__head_subtitle_decls_tl
  , quote-decls    = \l__head_quote_decls_tl
```

```
  , number-format  = \__head_number_format:n
  , contents-extra = \l__head_contents_extra_tl
}
{
  \__head_show_arguments:nnnnnnnnn
       {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
```

store the value of `secnumdepth` so that we can change it locally.

```
  \tl_set:Ne\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}
```

define `\theheading`

```
  \tl_set_eq:Nc \theheading { the \l__head_name_tl }
  \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{runin}{#1} }
  \tl_if_empty:oT \l__head_start_code_tl
   {
    \str_case:Vn \l__head_placement_tl
      {
        { top }   { \tl_set:Nn \l__head_start_code_tl {\clearpage } }
      }
```

All other cases want an empty `\l__head_start_code_tl` so nothing to do.

```
%      { \tl_clear:N \l__head_start_code_tl }
   }
%
```

Nothing at all to do (for now) for `\l__head_final_code_tl`, it should by default be empty in all heading placement. But maybe we end up supporting further placements, so ...

```
%  \tl_if_empty:oT \l__head_final_code_tl
%   { \str_case:VnF \l__head_placement_tl
%      {
%        { top } { \tl_clear:N \l__head_final_code_tl }
%      }
%      { \tl_clear:N \l__head_final_code_tl }
%   }
  \__head_determine_penalty:
  \__head_determine_number_typesetting:N #2
  \__head_vertical_before_spacing:
```

We are in vmode now and here is the point where we (with tagging) can close a previous Sect structure and open the new one. We also have to initialize the change in the paratagging here as run-in titles typeset the heading in everypar.

```
  \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
  \UseTaggingSocket{sec/begin}
    {{\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
  \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
  \def \@svsechd {
      \__head_debug_typeout:n{use~ 'headformat'~instance:~
                              \l__head_headformat_instance_tl }
      \use:e {
        \UseInstance{headformat} { \l__head_headformat_instance_tl }
                 { \exp_not:o \UnusedTemplateKeys }
                 { \exp_not:o { \l__head_typeset_number_tl } }
                 { \exp_not:n { #3 } }
```

```
                { \exp_not:o { \use_i:nn  #9 } }
                { \exp_not:o { \use_ii:nn  #9 } }
      }
      \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
```

Restore `secnumdepth`

```
      \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
  }
%
  \@nobreakfalse
  \global\@noskipsectrue
  \everypar{%
    \if@noskipsec
       \global\@noskipsecfalse
       {\setbox\z@\lastbox}
       \clubpenalty\@M
%FMi group in headformat
%      \begingroup
       \@svsechd
%      \endgroup
       \unskip
```

This tagging socket starts the "paragraph" after the run-in heading

```
       \UseTaggingSocket{sec/title/split}
       \skip_horizontal:N \l__head_after_skip
    \else
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi
  }
```

— prepare next paragraph (does nothing by default)

```
  \l__head_final_code_tl
%
  \ignorespaces
}
```

11.6.5 headformat templates code

`headformat display` (*templ.*) This template creates a headformat where the number is on a line on its own.

```
\DeclareTemplateCode{headformat}{display}{5}
 {
  , indent        = \l__head_indent_dim
  , before-code   = \l__head_before_code_tl
  , after-code    = \l__head_after_code_tl
  , number-title-sep =  \l__head_number_title_sep_tl
 }
 {
  \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}
```

First evaluate any key setting done by the user (normally supplied from the main heading isntance.

```
  \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{display}{#1} }
  \group_begin:
  \UseTaggingSocket{sec/title/begin}{{\int_use:N\l__head_level_int}{#3}}
```

```
    \normalfont \normalcolor
    \interlinepenalty \@M
    \l__head_decls_tl{}
```

If there is a number and so a prefix, the link target should be before the number. We use for now the same place in the unnumbered case. TODO: If we put the target here we do not know the height of the line and the target is perhaps not high enough. And for unnumbered chapter it is perhaps too high. Check!

```
    \bool_if:NTF \l__head_unnumbered_bool
      {
        \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
          {
            \skip_horizontal:N \l__head_indent_dim
            \MakeLinkTarget[\l__head_name_tl]{}
          }
          {
            \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_indent_dim
          }
      }
      {
        \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
          {
            \skip_horizontal:N \l__head_indent_dim
            \MakeLinkTarget{\l__head_name_tl}
          }
          {
            \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
          }
        \l__head_number_decls_tl
        #2
        \par\nobreak
        \skip_vertical:n { \l__head_number_title_sep_tl }
      }
    \l__head_title_decls_tl

    \l__head_before_code_tl
    {#3}
    \l__head_after_code_tl
    \par
    \UseTaggingSocket{sec/title/end}
  \group_end:
}
```

probably better to use a format:n and drop these two code variables even though they are used by titlesec

`headformat hang` (*templ.*) This template creates a heading with a hanging number.

```
\DeclareTemplateCode{headformat}{hang}{5}
 {
  , indent        = \l__head_indent_dim
  , before-code   = \l__head_before_code_tl
  , after-code    = \l__head_after_code_tl
  , number-title-sep =  \l__head_number_title_sep_tl
 }
 {
  \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}
```

First evaluate any key setting done by the user (normally supplied from the main heading isntance.

```
\tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }

\group_begin:
  \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}

  \normalfont \normalcolor
  \interlinepenalty \@M
  \l__head_decls_tl{}
```

The link target should be at the left text margin, or, if the section is moved into the margin, at the left of the number.

```
  \bool_if:NTF \l__head_unnumbered_bool
    {
      \tl_set:Nn\l__head_tmpa_tl
       {
        \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
          {
              \skip_horizontal:N \l__head_indent_dim \MakeLinkTarget[\l__head_name_tl]{}
          }
          {
              \MakeLinkTarget[\l__head_name_tl]{}
              \skip_horizontal:N \l__head_indent_dim
          }
       }
    }
    {
     \tl_set:Nn \l__head_tmpa_tl
      {
        \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
            {
              \skip_horizontal:N \l__head_indent_dim  \MakeLinkTarget{\l__head_name_tl}
            }
            {
              \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
            }
            { \l__head_number_decls_tl #2 }
            \skip_horizontal:n { \l__head_number_title_sep_tl }
       }
     }
   \UseTaggingSocket{sec/title/hang}
     {{\int_use:N\l__head_level_int}\l__head_unnumbered_bool{\l__head_tmpa_tl}{#3}}
     {
      \@hangfrom
        {
         \l__head_tmpa_tl
        }
     }
  \l__head_title_decls_tl

  \l__head_before_code_tl
  {#3}
  \l__head_after_code_tl
  \par
```

probably better to use a format:n and drop these two code variables even though they are used by titlesec

```
  \group_end:
}
```

`headformat runin` (*templ.*) This template creates a heading with a run-in title. Arguments are key-value, number, title, subtitle, quotation.

```
\DeclareTemplateCode{headformat}{runin}{5}
{
 , indent        = \l__head_indent_dim
 , before-code   = \l__head_before_code_tl
 , after-code    = \l__head_after_code_tl
 , number-title-sep =  \l__head_number_title_sep_tl
}
{
 \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}
```

First evaluate any key setting done by the user (normally supplied from the main heading isntance.

```
  \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }
  \group_begin:
    \normalfont \normalcolor
    \interlinepenalty \@M
    \l__head_decls_tl{}
```

Setting `\interlinepenalty` makes little sense I think (but that's the way it was in LaTeX 2_ε)

We must avoid that the sep between number and title is used in the unnumbered case. So we test with the boolean.

```
    \bool_if:NTF \l__head_unnumbered_bool
      {
        \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
         {
           \skip_horizontal:N \l__head_indent_dim
           \MakeLinkTarget[\l__head_name_tl]{}
         }
         {
           \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_indent_dim
         }
      }
      {
         \dim_compare:nNnTF  \l__head_indent_dim < \c_zero_skip
         {
           \skip_horizontal:N \l__head_indent_dim \MakeLinkTarget{\l__head_name_tl}
         }
         {
           \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
         }
         {
          \l__head_number_decls_tl
          \UseTaggingSocket{sec/title/number}{\int_use:N\l__head_level_int}{#2}
         }
         \skip_horizontal:n { \l__head_number_title_sep_tl }
      }
    \l__head_title_decls_tl
```

UFi:this should probably be a format command.

```
    \l__head_before_code_tl
    {#3}
    \l__head_after_code_tl
  \group_end:
}
```

11.6.6 Internal commands used by the template code

__head_determine_penalty:

```
\cs_new:Npn \__head_determine_penalty: {
```

If a penalty was specified use it, otherwise use `\@secpenalty`.

```
  \int_compare:nNnT \l__head_penalty_int = \c_max_int
    { \int_set:Nn \l__head_penalty_int \@secpenalty }
}
```

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

__head_determine_number_typesetting:N

```
\cs_new:Npn \__head_determine_number_typesetting:N #1 {
```

Using or suppressing a heading number depends on the heading level compared to the document value of `\c@secnumdepth`. If that doesn't suppress the number then an explicit key or a star form might still have suppressed it.

```
  \bool_set:Nn \l__head_unnumbered_bool
     { \bool_lazy_or_p:nn
       { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
       { \bool_if_p:N #1 }
     }
```

If we aren't producing a heading with a number we set `\c@secnumdepth` to a low number (it is reset at the end of the template code)

```
  \bool_if:NTF \l__head_unnumbered_bool
    {
      \int_gset:Nn\c@secnumdepth{-99}
```

and we set `\l__head_typeset_number_tl` to do nothing.

```
      \tl_clear:N \l__head_typeset_number_tl
    }
```

Otherwise the heading counter is incremented and a formatted version of the number plus any following (or preceding) space is stored in `\l__head_typeset_number_tl`. We use the kernel version of `\refstepcounter` as anchors are handled elsewhere.

```
    {
      \@kernel@refstepcounter{ \l__head_name_tl }
      \protected@edef \l__head_typeset_number_tl
        {
          \__head_number_format:n { \l__head_name_tl }
        }
    }
}
```

(*End of definition for* `\__head_determine_number_typesetting:N`.)

__head_vertical_before_spacing:

```
\cs_new:Npn \__head_vertical_before_spacing: {
  \tl_if_blank:VTF \l__head_start_code_tl
     {
       \if@noskipsec \leavevmode \fi
       \par
       \if@nobreak
         \everypar{}
       \else
         \addpenalty \l__head_penalty_int
         \addvspace  \l__head_before_skip
```

The \vspace* inserts a rule, we insert therefore the skip only if it is different to zero.

```
         \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
          { \vspace*    \l__head_after_penalty_skip }
       \fi
     }
     {
       \l__head_start_code_tl
```

If \l__head_start_code_tl holds code, we assume that it handles pagination, e.g., a \clearpage, etc. We therefore only add the skip that would follow the penalty.

```
       \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
          { \vspace*    \l__head_after_penalty_skip }
     }
}
```

(*End of definition for* __head_vertical_before_spacing:.)

\addcontentslinebookmark
\addcontentslinebookmarkOff
\addcontentslinebookmarkOn

We want to be able to control bookmarks independently from the toc entries, and also want to disable a bookmark by setting it to empty. For this we need some command to handle the bookmark command.

```
\newcommand\addcontentslinebookmark[3]{}
\newcommand\addcontentslinebookmarkOff{}
\newcommand\addcontentslinebookmarkReset{}
\providecommand\texorpdfstring[2]{#1}
\AddToHook{package/hyperref/after}
 {
  \RenewCommandCopy\addcontentslinebookmark\Hy@addcontentsline@addbookmark
  \renewcommand\addcontentslinebookmarkOff
   {
    \ifHy@bookmarks
     \let\addcontentslinebookmarkReset\Hy@bookmarkstrue
    \else
      \let\addcontentslinebookmarkReset\relax
    \fi
    \Hy@bookmarksfalse
   }
 }
```

(*End of definition for* \addcontentslinebookmark, \addcontentslinebookmarkOff, *and* \addcontentslinebookmarkOn. *These functions are documented on page* ??.)

__head_handle_marks_etc:nnnnn

Arg 1: toc entry, arg 2: mark, arg 3: bookmark, arg 4: nameref, arg 5: label code

```
\cs_new:Npn \__head_handle_marks_etc:nnnnn #1#2#3#4#5 {
%
```

```
    \tl_set:Nn\@currentlabelname{#4}
    \IfBlankF {#2}
       { \__head_mark_cmd:n { #2 } }
    \IfBlankTF {#1}
```

If the toc entry is empty the bookmarks must be set without `\addcontentsline`

```
      {
        \IfBlankF{#3}
         {
          \addcontentslinebookmark{toc}{\l__head_name_tl}
            {
              \bool_if:NF \l__head_unnumbered_bool
                {
                  \protect\numberline{ \use:c{ the \l__head_name_tl } }
                }
              #3
            }
         }
      }
      {
```

If the bookmark entry is empty we must disable the bookmarks locally before the `\addcontentsline` and reenable afterwards. We do not check if they are already disabled, perhaps later.

```
        \IfBlankT{#3}{\addcontentslinebookmarkOff}
        \addcontentsline{toc}{ \l__head_name_tl }
           {
             \bool_if:NF \l__head_unnumbered_bool
               {
                 \protect\numberline{ \use:c{ the \l__head_name_tl } }
               }
```

We use `\texorpdfstring` to separate toc and bookmark text.

```
             \texorpdfstring{#1}{#3}
           }
        \addcontentslinebookmarkReset
      }
```

perhaps this should have a hook for packages

Some headings (like `\chapter`) also want to write stuff to other contents files like `.lot` or `.lof`.

```
    \l__head_contents_extra_tl
```

We can always run the label code (it might be empty)

```
    \__head_debug_typeout:n{--->~label(s):~ \exp_not:n{#5}}
    #5
}
```

(*End of definition for* `\__head_handle_marks_etc:nnnnn`.)

11.7 Support for legacy classes and packages

If we define heading commands in the kernel (or in the tagging code) using

```
\DeclareDocumentCommand \section {s ={shorttitle}o m}
    { \ParseLaTeXeHeading {section} {#1} {#2} {#3} }
```

then this gets overwritten by every class right now.

If we redeclare them after the class was loaded then we overwrite the layout of legacy classes (done, for example, with `\@startsection`). New classes that use the above interface would be fine though, as long as we have declared the instances before the class is loaded.

So to make legacy classes work with their existing layout, we should not (ever) overwrite `\section` but let the class definition call `\@startsection` which would then set up suitable instances and only after that call `\ParseLaTeXeHeading`.

However, that would mean a "new" class like ltx-article would need to add the above definition and declare or edit the corresponding instances, instead of just declaring or adding the instances.

A perhaps better alternative could be to delay the definition of `\section` and friends until after the class got loaded and then take a peak at `\section` as defined by the class and if it contains a `\@startsection` call, leave it alone and otherwise overwrite it. And if the class hasn't defined `\section` (because it is a new class) declare it. Of course that means `\section` would then be available with every class even if it was never meant to contain headings.

But while writing this up, I start to think it is best if a new class defines both the document interface (i.e., the above command) as well as the layout (declare the instances) and the kernel does neither. It has been this way before and it is consistent, so why change.

The situation with headings defined via `\secdef` is worse: we don't have a nice handle as with `\@startsection` so there is no real way other than through static analysis to set up such a heading in the new template style. So all that is possible (I think) is that after the class has been loaded, we look if the usual candidates (`\part` and `\chapter`) have been defined and overwrite them with a standard layout. That would make the class tagging aware but, of course, would likely change the layout.

The current implementation

- provides instance for the heading commands of the standard classes;
- declares the heading commands for the standard classes with the new interfaces through class hooks;
- other classes call the adapted `\@startsection`; other headings command like `\part` or `\chapter` are not handled and must be setup individually.

11.7.1 Instances (sample/default definitions)

`heading part` (*inst.*) An instance suitable for book and report. An instance suitable for article is above in the hook.

```
\DeclareInstance{heading}{part}{display}
{
  , name          = part
  , level         = -1
  , placement         = page
  , after-penalty-sep = 0pt
  , number-format = \partname\nobreakspace\thepart
  , decls         = \centering\bfseries
  , number-decls  = \huge
  , title-decls   = \Huge
  , headformat-instance = display
```

```
  , mark-cmd      = \partmark {#1}
}
```

heading chapter (*inst.*)

```
\DeclareInstance{heading}{chapter}{display}
{
  , name          = chapter
  , level         = 0
  , placement         = top
  , after-penalty-sep = 50pt
%  , number-title-sep  = 20pt  % currently in headformat
  , after-sep     = 40pt
  , number-format = \@chapapp\space \thechapter
  , decls         = \raggedright \parindent0pt \bfseries
  , number-decls  = \huge
  , title-decls   = \Huge
  , headformat-instance = display
  , mark-cmd = \chaptermark {#1}
  , contents-extra=  \addtocontents{lof}{\protect\addvspace{10\p@}}%
                     \addtocontents{lot}{\protect\addvspace{10\p@}}%
}
```

heading section (*inst.*)

```
\DeclareInstance{heading}{section}{display}
{
  , name     = section
  , level    = 1
  , mark-cmd = \sectionmark {#1}
  , before-sep    = 3.5ex plus 1ex minus .2ex
  , after-sep     = 2.3ex plus .2ex
  , decls         = \normalfont\Large\bfseries
}
```

heading subsection (*inst.*)

```
\DeclareInstance{heading}{subsection}{display}
{
  , name  = subsection
  , parent-name = section
  , level = 2
  , mark-cmd = \subsectionmark {#1}
  , before-sep = 3.25ex plus 1ex minus .2ex
  , after-sep  = 1.5ex plus .2ex
  , decls      = \normalfont\large\bfseries
}
```

heading subsubsection (*inst.*)

```
\DeclareInstance{heading}{subsubsection}{display}
{
  , name  = subsubsection
  , parent-name = subsection
  , level = 3
  , before-sep = 3.25ex plus 1ex minus .2ex
  , after-sep  = 1.5ex plus .2ex
```

```
  , decls      = \normalfont\normalsize\bfseries
}
```

heading paragraph (*inst.*)

```
\DeclareInstance{heading}{paragraph}{runin}
{
  , name  = paragraph
  , parent-name = subsubsection
  , level = 4
  , before-sep = 3.25ex \@plus1ex \@minus.2ex
  , after-sep  = 1em
  , decls      = \normalfont\normalsize\bfseries
  , mark-cmd   =  \paragraphmark {#1}
  , headformat-instance = paragraph

}
```

heading subparagraph (*inst.*)

```
\DeclareInstance{heading}{subparagraph}{runin}
{
  , name  = subparagraph
  , parent-name = paragraph
  , level = 5
  , before-sep = 3.25ex \@plus1ex \@minus .2ex
  , after-sep  = 1em
  , decls      = \normalfont\normalsize\bfseries
  , mark-cmd   =  \subparagraphmark {#1}
  , headformat-instance = subparagraph

}
```

headformat display (*inst.*)

```
\DeclareInstance{headformat}{display}{display}
{
  , indent        = 0pt
  , before-code   =
  , after-code    =
}
```

headformat hang (*inst.*)

```
\DeclareInstance{headformat}{hang}{hang}
{
  , indent        = 0pt
  , before-code   =
  , after-code    =
}
```

headformat paragraph (*inst.*)

```
\DeclareInstance{headformat}{paragraph}{runin}
{
  , indent        = 0pt
  , before-code   =
  , after-code    =
}
```

headformat subparagraph (*inst.*)

```
\DeclareInstance{headformat}{subparagraph}{runin}
{
  , indent        = \parindent
  , before-code   =
  , after-code    =
}
```

adformat section-special (*inst.*) A special headformat instance for testing. It can be used with `\section[headformat-instance=section-special]{bla}`

```
\DeclareInstance{headformat}{section-special}{hang}
{
  , indent = 4em
  , after-code = !
}
```

11.7.2 New \@startsection

\@startsection To support legacy classes that implement headings through a call to `\@startsection` we redefine this command to generate a heading instance from the arguments of `\@startsection` if it doesn't yet exist and then use this instance to typeset the heading. NOTE: To handle legacy definition like `{\normalfont\Large\bfseries\MakeUppercase}` in the last argument it copies this argument both to `decls` and to `before-code`.

```
\DeclareDocumentCommand \@startsection {mmmmmm s ={shorttitle}o m}{
```

If there already exists an instance `#1-@startsection` then arguments 2–6 are ignored and we simply call that instance. Otherwise we go through the process to set it up. This allows classes, packages and authors to create and overwrite definitions with `\@startsection`. But after a call to a command using the `\@startsection` the instances are created. It is therefore not possible to redefine the command after a first use or to create two commands using the same level, e.g. `\section` and `\specialsection`. Such setups should use the new interfaces to declare heading commands.

```
  \IfInstanceExistsF{heading}{#1-@startsection}{
    \__head_debug_typeout:n{Info:~ setting~ up~ instances~ for~
                            legacy~ \string\@startsection}
```

In the LaTeX 2_ε logic a negative #4 means we do not indent the following paragraph … It is okay to change any `em` or `ex` specifications to real `pt` values here, because this code is executed in the same place where `\@startsection` is normally executed and inside the original definitions such assignments happen as well, before there is any font change happening for #4 and #5.

```
    \@tempskipa #4\relax
    \@afterindenttrue
    \ifdim \@tempskipa <\z@
      \@tempskipa -\@tempskipa
      \@afterindentfalse
    \fi
```

… and a negative #5 means we should produce a runin heading.

```
    \@tempskipb #5\relax
    \ifdim \@tempskipb>\z@
      \use:e {
```

```
        \DeclareInstance{heading}{#1-@startsection}{display}{
         , name       = #1
         , level      = #2
         , mark-cmd   = \exp_not:c{#1mark} {##1}
         , before-sep = \the\@tempskipa
         , after-sep  = \the\@tempskipb
         , para-indent = \if@afterindent true \else false\fi
         , decls      = \exp_not:n{#6}
```

we also do the declarations in `before-code` to handle settings which ends, e.g., with `\MakeUppercase`,

```
         , headformat-instance = #1-@startsection
      }}
```

We can expect that the instance `#1-@startsection` is not set up by the user if `#1-@startsection` isn't, so no check.

```
      \DeclareInstance{headformat}{#1-@startsection}{hang}{indent = #3,before-code={#6}}
    \else
      \@tempskipb=-\@tempskipb
      \use:e {
       \DeclareInstance{heading}{#1-@startsection}{runin}{
         , name       = #1
         , level      = #2
         , mark-cmd   = \exp_not:c{#1mark} {##1}
         , before-sep = \the\@tempskipa
         , after-sep  = \the\@tempskipb
         , decls      = \exp_not:n{#6}
         , headformat-instance = #1-@startsection
      }}
      \DeclareInstance{headformat}{#1-@startsection}{runin}{indent = #3,before-
code={#6}}
    \fi
  }
  \ParseLaTeXeHeading {#1-@startsection}{#7}{#8}{#9}
}
```

(*End of definition for* `\@startsection`. *This function is documented on page* **??**.)

Some classes redefine `\@startsection` and then our redefinition is lost. So we reinstate it

```
\NewCommandCopy\kernel@startsection\@startsection
\AddToHook{class/after}[head/@startsection]
  {\RenewCommandCopy\@startsection\kernel@startsection}
```

11.7.3 New `\secdef`

The command maps standard `\secdef` definition of `\part` and `\chapter` to the new interface. Unknown heading commands warn (or error if tagging is active) and then call the old commands.

`\secdef`

```
\RenewDocumentCommand\secdef{mmsO{#5}m}
 {
  \str_case:enF {\cs_to_str:N#1}
   {
```

```
    {@part}
     {
       \IfNoValueTF{#4}
        {\ParseLaTeXeHeading{part}{#3} {start-code=\relax} {#5}}
        {
         \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
         \ExpandArgs{nne}\ParseLaTeXeHeading{part}{#3}{start-code=\relax,\exp_not:o{\l__head_
        }
       \@latex@warning{\noexpand\secdef~detected~in~\noexpand\part~command.\MessageBreak
         \noexpand\part~will~be~redefined~to~use~templates.\MessageBreak
         This~may~change~the~layout!}
       \DeclareDocumentCommand \part {s ={shorttitle}o m}
         { \ParseLaTeXeHeading {part} {##1} {##2} {##3} }
     }
    {@chapter}
     {
       \IfNoValueTF{#4}
        { \ParseLaTeXeHeading{chapter}{#3} {#4} {#5} }
        {
          \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
          \ExpandArgs{nno}\ParseLaTeXeHeading{chapter}{#3}{\l__head_tmpa_tl} {#5}
        }
       \@latex@warning{\noexpand\secdef~detected~in~\noexpand\chapter~command.\MessageBreak
         \noexpand\chapter~will~be~redefined~to~use~templates.\MessageBreak
         This~may~change~the~layout!}
       \DeclareDocumentCommand \chapter {s ={shorttitle}o m}
        { \ParseLaTeXeHeading {chapter} {##1} {##2} {##3} }
     }
   }
   {
    \tag_if_active:TF\@latex@error\@latex@warning
     {\noexpand\secdef~with~unknown~argument~\noexpand#1 found.\MessageBreak
      The~command~can~not~be~adapted~on~the~fly~to~support~tagging.\MessageBreak
      The~heading~command~using~this~\noexpand\secdef~should~be~
                  reimplemented\MessageBreak with~templates}{}
    \IfBooleanTF{#3}{#2{#5}}{#1[#4]{#5}}
   }
 }
```

A helper command to reprocess the argument as key-val

```
\NewDocumentCommand\__head_process_shorttitle:nN{={shorttitle}mm}
 { \tl_set:Nn#2{#1} }
```

(*End of definition for* `\secdef`. *This function is documented on page* **??**.)

11.7.4 Core LaTeX

We define here as an example the heading commands with the new interfaces for the three standard classes.

```
\AddToHook{class/article/after}[head/example]
 {
   \DeclareInstance{heading}{part}{display}
     {
      , name          = part
      , level         = -1
```

```
      , before-sep    = 4ex
      , after-sep     = 3ex
      , number-format = \partname\nobreakspace\thepart
      , decls         = \raggedright\bfseries
      , number-decls  = \Large
      , title-decls    = \huge
      , headformat-instance = part
      , mark-cmd      = \partmark {#1}
    }
  \DeclareInstance{headformat}{part}{display}
    {
      , indent        = 0pt
      , before-code   =
      , after-code    =
      , number-title-sep = 0pt
    }
  \DeclareDocumentCommand \part {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
  \__head_setup_default_heading:
}

\AddToHook{class/report/after}[head/example]
{
   \DeclareDocumentCommand \part {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
   \DeclareDocumentCommand \chapter {s ={shorttitle}o m}
     {
       \if@mainmatter
        \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
       \else
        \IfBooleanTF{#1}
         {% starred:
          \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
         }
         {\IfNoValueTF{#2}
           {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {shorttitle={#3}} {#3}}
           {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}
         }
       \fi
     }
   \__head_setup_default_heading:
}
\AddToHook{class/book/after}[head/example]
{
   \DeclareDocumentCommand \part {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
   \DeclareDocumentCommand \chapter {s ={shorttitle}o m}
     {
       \if@mainmatter
        \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
       \else
        \IfBooleanTF{#1}
         {% starred:
          \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
```

```
          }
          {\IfNoValueTF{#2}
            {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {shorttitle={#3}} {#3}}
            {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}
          }
        \fi
      }
    \__head_setup_default_heading:
}
\cs_new_protected:Npn \__head_setup_default_heading:
 {
```

\section

```
   \DeclareDocumentCommand \section {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {section} {##1} {##2} {##3} }
```

(*End of definition for* \section*. This function is documented on page* ??.)

\subsection

```
   \DeclareDocumentCommand \subsection {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {subsection} {##1} {##2} {##3} }
```

(*End of definition for* \subsection*. This function is documented on page* ??.)

\subsubsection

```
   \DeclareDocumentCommand \subsubsection {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {subsubsection} {##1} {##2} {##3}  }
```

(*End of definition for* \subsubsection*. This function is documented on page* ??.)

\paragraph

```
   \DeclareDocumentCommand \paragraph {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {paragraph} {##1} {##2} {##3} }
```

(*End of definition for* \paragraph*. This function is documented on page* ??.)

\subparagraph

```
   \DeclareDocumentCommand \subparagraph {s ={shorttitle}o m}
     { \ParseLaTeXeHeading {subparagraph} {##1} {##2} {##3} }
 } %end of command
```

(*End of definition for* \subparagraph*. This function is documented on page* ??.)

```
⟨/package⟩
```

12 Issues and problems noticed along the way

12.1 Local \DeclareInstance

\DeclareInstance does its declaration locally. That might be the right decision (or not) but we need to make sure that it in that case all of the declaration is local.

One potential problem with that is that it might allocate TEX registers.

The problem showed up in the redefinition of \@startsection, because in the current document some headings are local to an environment, so things get redeclared over and over again.

12.2 \SetCurrentTemplateKeys

Furthermore, I think I would like to have some `\SetCurrentTemplateKeys` where one does not have to specify the type nor the template name, because that is how it is always used (by me).

12.3 O-expansion of \UseInstance arguments

Whenever a template code calls a sub-instance and that sub-instance takes arguments, then the values for these arguments are typically inside tokenlists or registers so that for efficiency (and sometimes as a total must) one has to o-expand all arguments. While that can be coded somehow, e.g.,

```
\use:e {
  \UseInstance{headformat} { \l_@@_headformat_instance_tl }
              { \exp_not:o \UnusedTemplateKeys }
              { \exp_not:o { \l_@@_typeset_number_tl } }
              { \exp_not:n { #3 } }
              { \exp_not:o { \use_i:nn  #9 } }
              { \exp_not:o { \use_ii:nn  #9 } }
}
```

it would be better to have a version of `\UseInstance` that does this automatically. No suggestion for a name.

12.4 Switch \openright is not defined by core

I think this should change and it might be enough to just define it in the kernel (I think `\newif` no longer complains if it acts on an existing `\if...`

12.5 Indexheading at least for l3doc is wrong now

Needs checking.

```
⟨*latex-lab⟩
\ProvidesFile{sec-template-latex-lab-testphase.ltx}
        [\ltlabsecIIdate\space v\ltlabsecIIversion\space latex-lab wrapper sec-
template]
\RequirePackage{latex-lab-testphase-sec-template}
⟨/latex-lab⟩
```

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.

U

V