

# Reimplementation of $\text{\LaTeX}$ 2 $\epsilon$ 's heading commands using templates

$\text{\LaTeX}$  Project\*

v0.9a 2026-01-14

## Abstract

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Changes to the user interface</b>	<b>3</b>
2.1	New user commands . . . . .	3
2.2	The optional argument . . . . .	3
2.3	Class support . . . . .	4
2.4	Changing heading commands . . . . .	4
2.5	Converting heading commands defined with <code>\@startsection</code> . . . . .	5
2.5.1	Step 1: Getting the template name and the instance values . . . . .	5
2.5.2	Step 2: Get the default template values . . . . .	6
2.6	Step 3: Declare the instances . . . . .	7
2.6.1	Step 4 Redefine the <code>\section</code> command . . . . .	7
<b>3</b>	<b>Setting up <math>\text{\LaTeX}</math> 2<math>\epsilon</math> heading commands in classes</b>	<b>7</b>
<b>4</b>	<b>Open points</b>	<b>8</b>
<b>5</b>	<b>Template types and templates for headings</b>	<b>8</b>
5.1	Template types . . . . .	8
5.1.1	The template type ‘heading’ . . . . .	8
5.1.2	The template type ‘headformat’ . . . . .	9
5.2	Templates . . . . .	10
5.2.1	The <code>heading</code> template ‘ <code>\all</code> ’ . . . . .	10
5.2.2	The <code>heading</code> template ‘display’ . . . . .	11
5.2.3	The <code>heading</code> template ‘runin’ . . . . .	12
5.2.4	The <code>headformat</code> template ‘hang’ . . . . .	12
5.2.5	The <code>headformat</code> template ‘runin’ . . . . .	13
5.2.6	The <code>headformat</code> template ‘display’ . . . . .	13
5.3	Default instances . . . . .	14
5.3.1	Instances of <code>heading</code> templates . . . . .	14
5.3.2	Instances of <code>headformat</code> templates . . . . .	14

---

\*Initial implementation by Frank Mittelbach.

<b>6</b>	<b>Support for legacy classes and packages</b>	<b>15</b>
6.1	Support classes based on legacy L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> interfaces . . . . .	15
6.2	Support for the titlesec package interfaces . . . . .	15
<b>7</b>	<b>Support for links</b>	<b>15</b>
<b>8</b>	<b>Bookmarks</b>	<b>16</b>
<b>9</b>	<b>Tagging support</b>	<b>16</b>
<b>10</b>	<b>Debugging support</b>	<b>16</b>
<b>11</b>	<b>The Implementation</b>	<b>16</b>
11.1	Debugging . . . . .	16
11.2	Temp variable(s) . . . . .	18
11.3	variable(s) . . . . .	18
11.4	New user commands . . . . .	18
11.5	The L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> parsing interface . . . . .	19
11.6	Templates . . . . .	22
11.6.1	Template types . . . . .	22
11.6.2	heading templates interfaces . . . . .	22
11.6.3	headformat templates interfaces . . . . .	23
11.6.4	heading templates code . . . . .	24
11.6.5	headformat templates code . . . . .	29
11.6.6	Internal commands used by the template code . . . . .	33
11.7	Support for legacy classes and packages . . . . .	35
11.7.1	Instances (sample/default definitions) . . . . .	36
11.7.2	New \startsection . . . . .	39
11.7.3	New \secdef . . . . .	40
11.7.4	Core L <sup>A</sup> T <sub>E</sub> X . . . . .	41
<b>12</b>	<b>Issues and problems noticed along the way</b>	<b>43</b>
12.1	Local \DeclareInstance . . . . .	43
12.2	\SetCurrentTemplateKeys . . . . .	44
12.3	O-expansion of \UseInstance arguments . . . . .	44
12.4	Switch \openright is not defined by core . . . . .	44
12.5	Indexheading at least for l3doc is wrong now . . . . .	44
	<b>Index</b>	<b>45</b>

## 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** These 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 L<sup>A</sup>T<sub>E</sub>X 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[shorttitle=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 re-defined 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 => Opt
> 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 => Opt
> penalty => \c_max_int
> after-penalty-sep => Opt
> after-sep => Opt
> 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 => Opt
> 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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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 rudimentarily 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 `\NoValue`. If it is offered via a key value interface, the template receives whatever is set by the user (and `\NoValue` 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 `\@secntformat`, 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.<sup>1</sup>

### 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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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 certainly simpler from a compatibility perspective).

All templates should set up `\theheading` to correspond to `\the<name>`.

---

<sup>1</sup>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  $\text{T}_{\text{E}}\text{X}$  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  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  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  $\text{T}_{\text{E}}\text{X}$  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  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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 L<sup>A</sup>T<sub>E</sub>X'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:
11 \cs\_new\_protected:Npn \head\_debug\_on:
12 {
13   \bool\_gset\_true:N \g\_head\_debug\_bool
14   \\_head\_debug\_gset:
15 }
16 \cs\_new\_protected:Npn \head\_debug\_off:
17 {
18   \bool\_gset\_false:N \g\_head\_debug\_bool
19   \\_head\_debug\_gset:
20 }
21 \cs\_new\_protected:Npn \\_head\_debug\_gset:
22 {
23   \cs\_gset\_protected:Npx \\_head\_debug:n ##1
24   { \bool\_if:NT \g\_head\_debug\_bool {##1} }
25   \cs\_gset\_protected:Npx \\_head\_debug\_typeout:n ##1
26   { \bool\_if:NT \g\_head\_debug\_bool { \typeout{[head]~ ##1} } }
27 }

```

(End of definition for `\head\_debug\_on:`, `\head\_debug\_off:`, and `\_head\_debug\_gset:`. These functions are documented on page 16.)

`\DebugHeadingsOn`  
`\DebugHeadingsOff`

```

28 \cs\_new\_protected:Npn \DebugHeadingsOn { \head\_debug\_on: }
29 \cs\_new\_protected:Npn \DebugHeadingsOff { \head\_debug\_off: }
30 \DebugHeadingsOff

```

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

`\_head\_show\_arguments:nnnnnn`

Some debug data ...

```

31 \cs\_new:Npn \_head\_show\_arguments:nnnnnn #1#2#3#4#5 {
32   \_head\_debug\_typeout:n{-----~ Headformat~ instance~ arguments:}
33   \_head\_debug\_typeout:n{1:~ keys~ =~ \exp\_not:n{#1}}
34   \_head\_debug\_typeout:n{2:~ number~ =~ \exp\_not:n{#2}}
35   \_head\_debug\_typeout:n{3:~ title~ =~ \exp\_not:n{#3}}
36   \_head\_debug\_typeout:n{4:~ subtitle~ =~ \exp\_not:n{#4}}
37   \_head\_debug\_typeout:n{5:~ quotation~ =~ \exp\_not:n{#5}}
38 }

```

(End of definition for `\_head\_show\_arguments:nnnnnn`.)

`\_head\_show\_arguments:nnnnnnnnnn`

Some debug data ...

```

39 \cs\_new:Npn \_head\_show\_arguments:nnnnnnnnnn #1#2#3#4#5#6#7#8#9 {
40   \_head\_debug\_typeout:n{-----~ Heading~ instance~ arguments:}
41   \_head\_debug\_typeout:n{1:~ keys~ =~ \exp\_not:n{#1}}
42   \_head\_debug\_typeout:n{2:~ unnumbered~ =~ \exp\_not:n{#2}}
43   \_head\_debug\_typeout:n{3:~ title~ =~ \exp\_not:n{#3}}
44   \_head\_debug\_typeout:n{4:~ toc~ =~ \exp\_not:n{#4}}
45   \_head\_debug\_typeout:n{5:~ running~ =~ \exp\_not:n{#5}}
46   \_head\_debug\_typeout:n{6:~ bookmark~ =~ \exp\_not:n{#6}}
47   \_head\_debug\_typeout:n{7:~ nameref~ =~ \exp\_not:n{#7}}
48   \_head\_debug\_typeout:n{8:~ label~ =~ \exp\_not:n{#8}}

```

```

49 \l__head_debug_typeout:n{9:~ subtitle | quote ~ =~ \exp_not:n{#9}}
50 }

```

(End of definition for \l\_\_head\_show\_arguments:nnnnnnnnn.)

## 11.2 Temp variable(s)

```
\l__head_tmpa_tl
```

```
51 \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.

```

52 %\tl_new:N \l__head_bookmark_tl
53 %\tl_new:N \l__head_running_tl
54 %\tl_new:N \l__head_toc_tl
55 %\tl_new:N \l__head_nameref_tl
56 %\tl_new:N \l__head_subtitle_tl
57 %\tl_new:N \l__head_quote_tl
58 %\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:

```

59 \tl_new:N\l__head_label_tl
60 \tl_new:N\l__head_instance_keys_tl
61 \tl_new:N\l__head_typeset_number_tl
62 \tl_new:N\l__head_saved_secnumdepth_tl
63 \tl_new:N \l__head_title_tl
64 \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.

```
65 \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.

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

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

## 11.5 The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> parsing interface

L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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).

```

67 \cs_new_protected:Npn \ParseLaTeXeHeading #1 #2 #3 #4 {
68   \__head_debug_typeout:n{=====}
69   \__head_debug_typeout:n{#1:~ \IfBooleanT{#2}{*}
70     \IfValueT{#3}{[\exp_not:n{#3}]}
71     {\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`.

```

72   \__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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> logic).

```

73   \IfBooleanTF{#2}
74   {
75     \tl_clear:N \l__head_toc_tl
76     \tl_clear:N \l__head_running_tl
77     \tl_clear:N \l__head_bookmark_tl
78     \bool_set_true:N \l__head_unnumbered_bool
79   }
80   {
81     \tl_set_eq:NN \l__head_toc_tl      \l__head_title_tl
82     \tl_set_eq:NN \l__head_running_tl  \l__head_title_tl
83     \tl_set_eq:NN \l__head_bookmark_tl \l__head_title_tl
84     \bool_set_false:N \l__head_unnumbered_bool
85   }

```

The `\nameref` always starts out matching the main title.

```

86   \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`

```
87 \tl_set_eq:NN \l__head_subtitle_tl \c_novalue_tl
88 \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?

```
89 \IfNoValueTF { #3 }
90 { \tl_clear:N \l__head_instance_keys_tl }
91 { \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.<sup>2</sup>

```
92 \__head_debug_typeout:n{use~ 'heading'~ instance:~ #1 }
93 \use:e {
94   \exp_not:N \UseInstance{heading}{#1}
95   { \exp_not:o \l__head_instance_keys_tl }
96   { \bool_if:NTF \l__head_unnumbered_bool \BooleanTrue \BooleanFalse }
97   { \exp_not:o \l__head_title_tl }
98   { \exp_not:o \l__head_toc_tl }
99   { \exp_not:o \l__head_running_tl }
100  { \exp_not:o \l__head_bookmark_tl }
101  { \exp_not:o \l__head_nameref_tl }
102  { \exp_not:o \l__head_label_tl }
103  { { \exp_not:o \l__head_subtitle_tl }
104    { \exp_not:o \l__head_quote_tl } }
105 }
106 }
```

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.

```
107 \keys_define:nn {heading} {
108   , shorttitle .meta:n =
109     {toc = {#1} , running = {#1} , bookmark = {#1} , nameref= {#1} }
110   , bookmark .tl_set:N = \l__head_bookmark_tl
111   , running .tl_set:N = \l__head_running_tl
112   , toc .tl_set:N = \l__head_toc_tl
113   , nameref .tl_set:N = \l__head_nameref_tl
114   %
115   , numbered .bool_set_inverse:N = \l__head_unnumbered_bool
116   , numbered .default:n = true
117   , unnumbered .bool_set:N = \l__head_unnumbered_bool
118   , unnumbered .default:n = true
119   %
```

---

<sup>2</sup>I think this is a common requirement and perhaps `\UseInstance` should really o-expand all arguments it receives.

```

120 , subtitle .tl_set:N = \l__head_subtitle_tl
121 , 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.

```

122 , label .code:n = \tl_put_right:Nn \l__head_label_tl { \label{#1} }
123 }

```

*(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.

```

124 \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.

```

125   \quark_if_no_value:nTF {#2}
126   {
127     \__head_debug_typeout:n{---no~label }
128     \tl_clear:N \l__head_label_tl
129     \tl_set:Nn\l__head_title_tl {#1#3}
130   }

```

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

```

131   {
132     \__head_debug_typeout:n{---label~found~#2}
133     \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`.

```

134     \tl_set:Nn\l__head_title_tl {#1}
135     \__head_find_label_aux:w #3\__head_find_label_aux:w
136   }
137   \__head_debug_typeout:n{---~return:~ '\exp_not:o \l__head_title_tl' }
138 }

```

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.

```

139 \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.

```

140   \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.

```

141   \quark_if_no_value:nF {#2}
142   {
143     \__head_debug_typeout:n{---~ extra~ label~ '#2'~ found }
144     \tl_put_right:Nn \l__head_label_tl { \label{#2} }
145     \__head_find_label_aux:w #3\__head_find_label_aux:w
146   }
147 }

```

*(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 }

```
148 \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

```
149 \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.

```
150 \DeclareTemplateInterface{heading}{display}{9}
151 {
152   , name           : tokenlist
153   , parent-name    : tokenlist
154   , reset-counter  : tokenlist
155   , level          : integer = 0
```

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

```
156   , placement      : choice {page , top , normal } = normal
157   , mark-cmd       : function(1) =
158   %
159   % many more keys for layout settings are missing for now
160   , 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.

```
161   , before-sep     : skip = Opt
162   , penalty        : integer = \c_max_int
163   , after-penalty-sep : skip = Opt
164   , after-sep      : skip = Opt
```

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).

```

165 , start-code      : tokenlist =      % no default values!
166 , final-code     : tokenlist =      % no default values!
167 , decls          : tokenlist = \normalfont
168 , number-decls   : tokenlist =
169 , title-decls    : tokenlist =
170 , subtitle-decls : tokenlist =
171 , quote-decls    : tokenlist =
172 , headformat-instance : tokenlist = hang
173 , number-format   : function(1) = \theheading
174 , contents-extra  : tokenlist =
175 }

```

`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`.

```

176 \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

```

177 \DeclareTemplateInterface{headformat}{display}{5}
178 {
179 , indent          : length = Opt
180 , before-code     : tokenlist =
181 , after-code      : tokenlist =
182 , number-title-sep : tokenlist = 20pt
183 }

```

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

```

184 \DeclareTemplateInterface{headformat}{hang}{5}
185 {
186 , indent          : length = Opt
187 , before-code     : tokenlist =
188 , after-code      : tokenlist =
189 , number-title-sep : tokenlist = 1em
190 }

```

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

```

191 \DeclareTemplateInterface{headformat}{runin}{5}
192 {
193 , indent          : length = Opt
194 , before-code     : tokenlist =
195 , after-code      : tokenlist =
196 , number-title-sep : tokenlist = 1em
197 }

```

### 11.6.4 heading templates code

heading display (*templ.*)

```

198 \DeclareTemplateCode{heading}{display}{9}
199 {
200   , name           = \l__head_name_tl
201   , level          = \l__head_level_int
202   % next two not yet used
203   , parent-name    = \l__head_pname_tl
204   , 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.

```

205   , placement      = {
206     , page          = \__head_debug_typeout:n{ A~ page~ heading }
207                     \tl_set:Nn \l__head_placement_tl { page }
208     , top           = \__head_debug_typeout:n{ A~ top~ heading }
209                     \tl_set:Nn \l__head_placement_tl { top }
210     , normal        = \__head_debug_typeout:n{ A~ normal~ heading }
211                     \tl_set:Nn \l__head_placement_tl { normal }
212   }
213   , mark-cmd       = \__head_mark_cmd:n

```

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

```

214   , para-indent    = {
215     , true          = \@afterindenttrue
216     , false         = \@afterindentfalse
217   }
218   , before-sep     = \l__head_before_skip
219   , penalty        = \l__head_penalty_int
220   , after-penalty-sep = \l__head_after_penalty_skip
221   , after-sep      = \l__head_after_skip
222   , start-code     = \l__head_start_code_tl
223   , final-code     = \l__head_final_code_tl
224   , headformat-instance = \l__head_headformat_instance_tl
225   , decls          = \l__head_decls_tl
226   , number-decls   = \l__head_number_decls_tl
227   , title-decls    = \l__head_title_decls_tl
228   , subtitle-decls = \l__head_subtitle_decls_tl
229   , quote-decls    = \l__head_quote_decls_tl
230   , number-format  = \__head_number_format:n
231   , contents-extra = \l__head_contents_extra_tl
232 }
233 {

```

```

234   \tl_set_eq:Nc \theheading { the \l__head_name_tl }

```

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

```

235   \tl_set:Nc \l__head_saved_secnumdepth_tl {\int_use:N \c@secnumdepth}
236   \__head_show_arguments:nnnnnnnnn
237   {#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.

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

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

```
238 \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.

```
239 \tl_if_empty:oT \l__head_start_code_tl
240 { \str_case:VnF \l__head_placement_tl
241   {
242     { page }
243     { \tl_set:Nn \l__head_start_code_tl
```

Straight from the placement definition of `\part`.

```
244   {
245     \if@openright
246       \cleardoublepage
247     \else
248       \clearpage
249     \fi
250     \thispagestyle{plain}%
251     \if@twocolumn
252       \onecolumn
253       \@tempswatrue
254     \else
255       \@tempswafalse
256     \fi
257     \null\vfil
258   }
259 }
260 { top }
261 { \tl_set:Nn \l__head_start_code_tl
262   {
263     \if@openright\cleardoublepage\else\clearpage\fi
264     \thispagestyle{plain}%
265     \global\@topnum\z@
266   }
267 }
268 }
269 { \tl_clear:N \l__head_start_code_tl }
270 }
271 %
272 \tl_if_empty:oT \l__head_final_code_tl
273 { \str_case:VnF \l__head_placement_tl
274   {
275     { page }
276     { \tl_set:Nn \l__head_final_code_tl
277       {
278         \vfil\newpage
279         \if@twoside
280           \if@openright
281             \null
282             \thispagestyle{empty}%
283           \newpage
284         \fi
```

```

285         \fi
286         \if@tempswa
287             \twocolumn
288         \fi
289     }
290 }
291 }
292 { \tl_set:Nn \l__head_final_code_tl { \@afterheading } }
293 }

```

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

```

294 \__head_determine_penalty:
295 \__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.

```

296 \__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.

```

297 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
298 \UseTaggingSocket{sec/begin}
299 {{\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

```

300 \__head_debug_typeout:n{use~ 'headformat'~instance:~
301                     \l__head_headformat_instance_tl }
302 \use:e {
303     \UseInstance{headformat} { \l__head_headformat_instance_tl }
304     { \exp_not:o \UnusedTemplateKeys }
305     { \exp_not:o { \l__head_typeset_number_tl } }
306     { \exp_not:n { #3 } }
307     { \exp_not:o { \use_i:nn #9 } }
308     { \exp_not:o { \use_ii:nn #9 } }
309 }
310 %
311 % --- handle marks, toc-entry, bookmark, nameref, and label
312 %
313 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
314 %
315 % --- post-heading handling (vertical)

```

Restore secnumdepth

```

316 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
317 \par \nobreak
318 \skip_vertical:N \l__head_after_skip
319 % --- prepare next paragraph (defaults to \cs{@afterheading}
320 %
321 \l__head_final_code_tl
322 %
323 \ignorespaces
324 }

```

```

325 \AddToHook{begindocument}{
326   \ifcsname if@openright\endcsname
327   \else

```

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

```

328   \expandafter\newif\csname if@openright\endcsname
329   \fi
330 }

```

(End of definition for .)

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

```

331 \DeclareTemplateCode{heading}{runin}{9}
332 {
333   , name          = \l__head_name_tl
334   , level         = \l__head_level_int
335   % next two not yet used
336   , parent-name   = \l__head_pname_tl
337   , 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.

```

338   , placement     = {
339     page          = \typeout{ ^^JA~ runin~ page~ placement~ heading~makes~no~sense
340                       (top-used)}
341     \tl_set:Nn \l__head_placement_tl { top }
342     ,top          = \__head_debug_typeout:n{ A~ top~ heading }
343     \tl_set:Nn \l__head_placement_tl { top }
344     ,normal       = \__head_debug_typeout:n{ A~ normal~ heading }
345     \tl_set:Nn \l__head_placement_tl { normal }
346   }
347   , 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.

```

348   , para-indent   = {
349     true          = %\typeout{para-indent~ setting~ ignored}
350     ,false        = %\typeout{para-indent~ setting~ ignored}
351   }
352   , before-sep    = \l__head_before_skip
353   , penalty       = \l__head_penalty_int
354   , after-penalty-sep = \l__head_after_penalty_skip
355   , after-sep     = \l__head_after_skip
356   , start-code    = \l__head_start_code_tl
357   , final-code    = \l__head_final_code_tl
358   , headformat-instance = \l__head_headformat_instance_tl
359   , decls         = \l__head_decls_tl
360   , number-decls  = \l__head_number_decls_tl
361   , title-decls   = \l__head_title_decls_tl
362   , subtitle-decls = \l__head_subtitle_decls_tl
363   , quote-decls   = \l__head_quote_decls_tl

```

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

```

364 , number-format = \_head\_number\_format:n
365 , contents-extra = \l\_head\_contents\_extra\_tl
366 }
367 {
368   \_head\_show\_arguments:nnnnnnnnn
369     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

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

```

370   \tl\_set:N\l\_head\_saved\_secnumdepth\_tl{\int\_use:N\c@secnumdepth}
define \theheading
371   \tl\_set\_eq:Nc \theheading { the \l\_head\_name\_tl }
372   \tl\_if\_empty:oF {#1} { \SetTemplateKeys{heading}{runin}{#1} }
373   \tl\_if\_empty:oT \l\_head\_start\_code\_tl
374   {
375     \str\_case:Vn \l\_head\_placement\_tl
376     {
377       { top } { \tl\_set:Nn \l\_head\_start\_code\_tl {\clearpage} } }
378     }

```

All other cases want an empty `\l\_head\_start\_code\_tl` so nothing to do.

```

379 %   { \tl\_clear:N \l\_head\_start\_code\_tl }
380 }
381 %

```

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 ...

```

382 %   \tl\_if\_empty:oT \l\_head\_final\_code\_tl
383 %   { \str\_case:VnF \l\_head\_placement\_tl
384 %     {
385 %       { top } { \tl\_clear:N \l\_head\_final\_code\_tl }
386 %     }
387 %     { \tl\_clear:N \l\_head\_final\_code\_tl }
388 %   }
389 \_head\_determine\_penalty:
390 \_head\_determine\_number\_typesetting:N #2
391 \_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.

```

392 \UseTaggingSocket{sec/end}{\int\_use:N\l\_head\_level\_int}
393 \UseTaggingSocket{sec/begin}
394   {\int\_use:N\l\_head\_level\_int}{tag=\UseStructureName{sec/\int\_use:N\l\_head\_level\_int}}
395 \UseTaggingSocket{sec/title/init}{\int\_use:N\l\_head\_level\_int}
396 \def \@svsechd {
397   \_head\_debug\_typeout:n{use~ 'headformat'~instance:~
398     \l\_head\_headformat\_instance\_tl }
399   \use:e {
400     \UseInstance{headformat} { \l\_head\_headformat\_instance\_tl }
401     { \exp\_not:o \UnusedTemplateKeys }
402     { \exp\_not:o { \l\_head\_typeset\_number\_tl } }
403     { \exp\_not:n { #3 } }

```

```

404         { \exp_not:o { \use_i:nn #9 } }
405         { \exp_not:o { \use_ii:nn #9 } }
406     }
407     \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
Restore secnumdepth
408     \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
409 }
410 %
411 \@nobreakfalse
412 \global\@noskipsectrue
413 \everypar{%
414     \if@noskipsec
415         \global\@noskipsecfalse
416         {\setbox\z@\lastbox}
417         \clubpenalty\@M
418 %Mi group in headformat
419 %     \begingroup
420 %     \@svsechd
421 %     \endgroup
422 %     \unskip
This tagging socket starts the “paragraph” after the run-in heading
423     \UseTaggingSocket{sec/title/split}
424     \skip_horizontal:N \l__head_after_skip
425 \else
426     \clubpenalty \@clubpenalty
427     \everypar{}%
428 \fi
429 }
— prepare next paragraph (does nothing by default)
430 \l__head_final_code_tl
431 %
432 \ignorespaces
433 }

```

### 11.6.5 headformat templates code

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

```

434 \DeclareTemplateCode{headformat}{display}{5}
435 {
436     , indent          = \l__head_indent_dim
437     , before-code     = \l__head_before_code_tl
438     , after-code      = \l__head_after_code_tl
439     , number-title-sep = \l__head_number_title_sep_tl
440 }
441 {
442     \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}

```

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

```

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

```

```

446 \normalfont \normalcolor
447 \interlinepenalty \@M
448 \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!

```

449 \bool_if:NTF \l__head_unnumbered_bool
450 {
451   \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
452   {
453     \skip_horizontal:N \l__head_indent_dim
454     \MakeLinkTarget[\l__head_name_tl]{}
455   }
456   {
457     \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_indent_dim
458   }
459 }
460 {
461   \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
462   {
463     \skip_horizontal:N \l__head_indent_dim
464     \MakeLinkTarget{\l__head_name_tl}
465   }
466   {
467     \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
468   }
469   \l__head_number_decls_tl
470   #2
471   \par\nobreak
472   \skip_vertical:n { \l__head_number_title_sep_tl }
473 }
474 \l__head_title_decls_tl
475 \l__head_before_code_tl
476 {#3}
477 \l__head_after_code_tl
478 \par
479 \UseTaggingSocket{sec/title/end}
480 \group_end:
481 }

```

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.

```

482 \DeclareTemplateCode{headformat}{hang}{5}
483 {
484   , indent           = \l__head_indent_dim
485   , before-code      = \l__head_before_code_tl
486   , after-code       = \l__head_after_code_tl
487   , number-title-sep = \l__head_number_title_sep_tl
488 }
489 {
490   \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}

```

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

```

491 \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }
492 \group_begin:
493 \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
494 \normalfont \normalcolor
495 \interlinepenalty \@M
496 \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.

```

497 \bool_if:NTF \l__head_unnumbered_bool
498 {
499   \tl_set:Nn\l__head_tmpa_tl
500   {
501     \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
502     {
503       \skip_horizontal:N \l__head_indent_dim \MakeLinkTarget[\l__head_name_tl]{}
504     }
505     {
506       \MakeLinkTarget[\l__head_name_tl]{}
507       \skip_horizontal:N \l__head_indent_dim
508     }
509   }
510 }
511 {
512   \tl_set:Nn \l__head_tmpa_tl
513   {
514     \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
515     {
516       \skip_horizontal:N \l__head_indent_dim \MakeLinkTarget{\l__head_name_tl}
517     }
518     {
519       \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
520     }
521     { \l__head_number_decls_tl #2 }
522     \skip_horizontal:n { \l__head_number_title_sep_tl }
523   }
524 }
525 \UseTaggingSocket{sec/title/hang}
526 {{\int_use:N\l__head_level_int}\l__head_unnumbered_bool{\l__head_tmpa_tl}{#3}}
527 {
528   \@hangfrom
529   {
530     \l__head_tmpa_tl
531   }
532 }
533 \l__head_title_decls_tl
534 \l__head_before_code_tl
535 {#3}
536 \l__head_after_code_tl
537 \par

```

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

```

538 \group_end:
539 }

```

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

```

540 \DeclareTemplateCode{headformat}{runin}{5}
541 {
542   , indent          = \l__head_indent_dim
543   , before-code     = \l__head_before_code_tl
544   , after-code      = \l__head_after_code_tl
545   , number-title-sep = \l__head_number_title_sep_tl
546 }
547 {
548   \__head_show_arguments:nnnnn {#1}{#2}{#3}{#4}{#5}

```

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

```

549 \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }
550 \group_begin:
551   \normalfont \normalcolor
552   \interlinepenalty \OM
553   \l__head_decls_tl{}

```

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

```

554 \bool_if:NTF \l__head_unnumbered_bool
555 {
556   \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
557   {
558     \skip_horizontal:N \l__head_indent_dim
559     \MakeLinkTarget[\l__head_name_tl]{}
560   }
561   {
562     \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_indent_dim
563   }
564 }
565 {
566   \dim_compare:nNnTF \l__head_indent_dim < \c_zero_skip
567   {
568     \skip_horizontal:N \l__head_indent_dim \MakeLinkTarget{\l__head_name_tl}
569   }
570   {
571     \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_indent_dim
572   }
573   {
574     \l__head_number_decls_tl
575     \UseTaggingSocket{sec/title/number}{\int_use:N\l__head_level_int}{#2}
576   }
577   \skip_horizontal:n { \l__head_number_title_sep_tl }
578 }
579 \l__head_title_decls_tl

```

Setting `\interlinepenalty` makes little sense I think (but that's the way it was in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>)



UFI: this should probably be a format command.

```

580 \l__head_before_code_tl
581 {#3}
582 \l__head_after_code_tl
583 \group_end:
584 }

```

### 11.6.6 Internal commands used by the template code

`\__head_determine_penalty:`

```

585 \cs_new:Npn \__head_determine_penalty: {
If a penalty was specified use it, otherwise use \@secpenalty.
586 \int_compare:nNnT \l__head_penalty_int = \c_max_int
587 { \int_set:Nn \l__head_penalty_int \@secpenalty }
588 }
(End of definition for \__head_determine_penalty:.)

```

`\__head_determine_number_typesetting:N`

```

589 \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.

```

590 \bool_set:Nn \l__head_unnumbered_bool
591 { \bool_lazy_or_p:nn
592 { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
593 { \bool_if_p:N #1 }
594 }

```

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)

```

595 \bool_if:NTF \l__head_unnumbered_bool
596 {
597 \int_gset:Nn \c@secnumdepth{-99}

```

and we set `\l__head_typeset_number_tl` to do nothing.

```

598 \tl_clear:N \l__head_typeset_number_tl
599 }

```

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.

```

600 {
601 \@kernel@refstepcounter{ \l__head_name_tl }
602 \protected@edef \l__head_typeset_number_tl
603 {
604 \__head_number_format:n { \l__head_name_tl }
605 }
606 }
607 }

```

*(End of definition for \\_\_head\_determine\_number\_typesetting:N.)*

`\_head\_vertical\_before\_spacing:`

```

608 \cs_new:Npn \_head\_vertical\_before\_spacing: {
609   \tl\_if\_blank:VTF \l\_head\_start\_code\_tl
610   {
611     \if@noskipsec \leavevmode \fi
612     \par
613     \if@nobreak
614       \everypar{}
615     \else
616       \addpenalty \l\_head\_penalty\_int
617       \addvspace \l\_head\_before\_skip

```

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

```

618       \dim\_compare:nNnF{\l\_head\_after\_penalty\_skip}={0pt}
619       { \vspace* \l\_head\_after\_penalty\_skip }
620     \fi
621   }
622   {
623     \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.

```

624       \dim\_compare:nNnF{\l\_head\_after\_penalty\_skip}={0pt}
625       { \vspace* \l\_head\_after\_penalty\_skip }
626     }
627   }

```

*(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.

```

628 \newcommand\addcontentslinebookmark[3]{}
629 \newcommand\addcontentslinebookmarkOff{}
630 \newcommand\addcontentslinebookmarkReset{}
631 \providecommand\texorpdfstring[2]{#1}
632 \AddToHook{package/hyperref/after}
633 {
634   \RenewCommandCopy\addcontentslinebookmark\Hy@addcontentsline@addbookmark
635   \renewcommand\addcontentslinebookmarkOff
636   {
637     \ifHy@bookmarks
638       \let\addcontentslinebookmarkReset\Hy@bookmarkstrue
639     \else
640       \let\addcontentslinebookmarkReset\relax
641     \fi
642     \Hy@bookmarksfalse
643   }
644 }

```

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

`\_head\_handle\_marks\_etc:nnnn`

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

```

645 \cs_new:Npn \_head\_handle\_marks\_etc:nnnnn #1#2#3#4#5 {
646   %

```

```

647 \tl_set:Nn\@currentlabelname{#4}
648 \IfBlankF {#2}
649   { \__head_mark_cmd:n { #2 } }
650 \IfBlankTF {#1}

```

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

```

651 {
652   \IfBlankF{#3}
653   {
654     \addcontentslinebookmark{toc}{\l__head_name_tl}
655     {
656       \bool_if:NF \l__head_unnumbered_bool
657       {
658         \protect\numberline{ \use:c{ the \l__head_name_tl } }
659       }
660       #3
661     }
662   }
663 }
664 {

```

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

```

665   \IfBlankT{#3}{\addcontentslinebookmarkOff}
666   \addcontentsline{toc}{ \l__head_name_tl }
667   {
668     \bool_if:NF \l__head_unnumbered_bool
669     {
670       \protect\numberline{ \use:c{ the \l__head_name_tl } }
671     }

```

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

```

672   \texorpdfstring{#1}{#3}
673 }
674 \addcontentslinebookmarkReset
675 }

```

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

```

676 \l__head_contents_extra_tl

```

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

```

677 \__head_debug_typeout:n{--->~label(s):~ \exp_not:n{#5}}
678 #5
679 }

```

(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} }

```

perhaps this should have  
a hook for packages

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.

```

680 \DeclareInstance{heading}{part}{display}
681 {
682   , name           = part
683   , level          = -1
684   , placement      = page
685   , after-penalty-sep = 0pt
686   , number-format  = \partname\nobreakspace\thepart
687   , decls          = \centering\bfseries
688   , number-decls   = \huge
689   , title-decls    = \Huge
690   , headformat-instance = display

```

```

691 , mark-cmd      = \partmark {#1}
692 }

```

heading chapter (*inst.*)

```

693 \DeclareInstance{heading}{chapter}{display}
694 {
695   , name          = chapter
696   , level         = 0
697   , placement     = top
698   , after-penalty-sep = 50pt
699   % , number-title-sep = 20pt % currently in headformat
700   , after-sep     = 40pt
701   , number-format = \@chapapp\space \thechapter
702   , decls         = \raggedright \parindent0pt \bfseries
703   , number-decls  = \huge
704   , title-decls   = \Huge
705   , headformat-instance = display
706   , mark-cmd      = \chaptermark {#1}
707   , contents-extra= \addtocontents{lof}{\protect\addvspace{10\p@}}%
708                   \addtocontents{lot}{\protect\addvspace{10\p@}}%
709 }

```

heading section (*inst.*)

```

710 \DeclareInstance{heading}{section}{display}
711 {
712   , name      = section
713   , level     = 1
714   , mark-cmd  = \sectionmark {#1}
715   , before-sep = 3.5ex plus 1ex minus .2ex
716   , after-sep  = 2.3ex plus .2ex
717   , decls     = \normalfont\Large\bfseries
718 }

```

heading subsection (*inst.*)

```

719 \DeclareInstance{heading}{subsection}{display}
720 {
721   , name      = subsection
722   , parent-name = section
723   , level     = 2
724   , mark-cmd  = \subsectionmark {#1}
725   , before-sep = 3.25ex plus 1ex minus .2ex
726   , after-sep  = 1.5ex plus .2ex
727   , decls     = \normalfont\large\bfseries
728 }

```

heading subsubsection (*inst.*)

```

729 \DeclareInstance{heading}{subsubsection}{display}
730 {
731   , name      = subsubsection
732   , parent-name = subsection
733   , level     = 3
734   , before-sep = 3.25ex plus 1ex minus .2ex
735   , after-sep  = 1.5ex plus .2ex

```

```

736 , decls      = \normalfont\normalsize\bfseries
737 }

```

heading paragraph (*inst.*)

```

738 \DeclareInstance{heading}{paragraph}{runin}
739 {
740   , name      = paragraph
741   , parent-name = subsubsection
742   , level     = 4
743   , before-sep = 3.25ex \@plus1ex \@minus.2ex
744   , after-sep  = 1em
745   , decls      = \normalfont\normalsize\bfseries
746   , mark-cmd   = \paragraphmark {#1}
747   , headformat-instance = paragraph
748 }
749 }

```

heading subparagraph (*inst.*)

```

750 \DeclareInstance{heading}{subparagraph}{runin}
751 {
752   , name      = subparagraph
753   , parent-name = paragraph
754   , level     = 5
755   , before-sep = 3.25ex \@plus1ex \@minus.2ex
756   , after-sep  = 1em
757   , decls      = \normalfont\normalsize\bfseries
758   , mark-cmd   = \subparagraphmark {#1}
759   , headformat-instance = subparagraph
760 }
761 }

```

headformat display (*inst.*)

```

762 \DeclareInstance{headformat}{display}{display}
763 {
764   , indent      = Opt
765   , before-code  =
766   , after-code   =
767 }

```

headformat hang (*inst.*)

```

768 \DeclareInstance{headformat}{hang}{hang}
769 {
770   , indent      = Opt
771   , before-code  =
772   , after-code   =
773 }

```

headformat paragraph (*inst.*)

```

774 \DeclareInstance{headformat}{paragraph}{runin}
775 {
776   , indent      = Opt
777   , before-code  =
778   , after-code   =
779 }

```

headformat subparagraph (*inst.*)

```

780 \DeclareInstance{headformat}{subparagraph}{runin}
781 {
782   , indent          = \parindent
783   , before-code     =
784   , after-code      =
785 }

```

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

```

786 \DeclareInstance{headformat}{section-special}{hang}
787 {
788   , indent = 4em
789   , after-code = !
790 }

```

### 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`.

```

791 \DeclareDocumentCommand \@startsection {mmmmm 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.

```

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

```

In the  $\text{\LaTeX} 2_{\epsilon}$  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`.

```

795     \@tempskipa #4\relax
796     \@afterindenttrue
797     \ifdim \@tempskipa <\z@
798       \@tempskipa -\@tempskipa
799       \@afterindentfalse
800     \fi

```

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

```

801     \@tempskipb #5\relax
802     \ifdim \@tempskipb >\z@
803       \use:e {

```

```

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

```

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

```

812   , headformat-instance = #1-@startsection
813 }

```

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

```

814 \DeclareInstance{headformat}{#1-@startsection}{hang}{indent = #3,before-code={#6}}
815 \else
816   \@tempskipb=-\@tempskipb
817   \use:e {
818     \DeclareInstance{heading}{#1-@startsection}{runin}{
819       , name      = #1
820       , level     = #2
821       , mark-cmd  = \exp_not:c{#1mark} {##1}
822       , before-sep = \the\@tempskipa
823       , after-sep  = \the\@tempskipb
824       , decls     = \exp_not:n{#6}
825       , headformat-instance = #1-@startsection
826     }
827     \DeclareInstance{headformat}{#1-@startsection}{runin}{indent = #3,before-
code={#6}}
828   \fi
829 }
830 \ParseLaTeXeHeading {#1-@startsection}{#7}{#8}{#9}
831 }

```

*(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

```

832 \NewCommandCopy\kernel@startsection\@startsection
833 \AddToHook{class/after}[head/@startsection]
834   {\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`

```

835 \RenewDocumentCommand\secdef{mmsO{#5}m}
836 {
837   \str_case:enF {\cs_to_str:N#1}
838   {

```



```

839   {@part}
840   {
841     \IfNoValueTF{#4}
842     {\ParseLaTeXeHeading{part}{#3} {start-code=\relax} {#5}}
843     {
844       \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
845       \ExpandArgs{nne}\ParseLaTeXeHeading{part}{#3}{start-code=\relax,\exp_not:o{\l__head_
846     }
847     \@latex@warning{\noexpand\secdef-detected-in~\noexpand\part~command.\MessageBreak
848     \noexpand\part~will~be~redefined~to~use~templates.\MessageBreak
849     This~may~change~the~layout!}
850     \DeclareDocumentCommand \part {s ={\shorttitle}o m}
851     { \ParseLaTeXeHeading {part} {##1} {##2} {##3} }
852   }
853   {@chapter}
854   {
855     \IfNoValueTF{#4}
856     { \ParseLaTeXeHeading{chapter}{#3} {#4} {#5} }
857     {
858       \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
859       \ExpandArgs{nnno}\ParseLaTeXeHeading{chapter}{#3}{\l__head_tmpa_tl} {#5}
860     }
861     \@latex@warning{\noexpand\secdef-detected-in~\noexpand\chapter~command.\MessageBreak
862     \noexpand\chapter~will~be~redefined~to~use~templates.\MessageBreak
863     This~may~change~the~layout!}
864     \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
865     { \ParseLaTeXeHeading {chapter} {##1} {##2} {##3} }
866   }
867 }
868 {
869   \tag_if_active:TF\@latex@error\@latex@warning
870   {\noexpand\secdef~with~unknown~argument~\noexpand#1 found.\MessageBreak
871   The~command~can~not~be~adapted~on~the~fly~to~support~tagging.\MessageBreak
872   The~heading~command~using~this~\noexpand\secdef~should~be~
873   reimplemented.\MessageBreak with~templates}{ }
874   \IfBooleanTF{#3}{#2{#5}}{#1[#4]{#5}}
875 }
876 }

```

A helper command to reprocess the argument as key-val

```

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

```

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

#### 11.7.4 Core L<sup>A</sup>T<sub>E</sub>X

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

```

879 \AddToHook{class/article/after}[head/example]
880 {
881   \DeclareInstance{heading}{part}{display}
882   {
883     , name          = part
884     , level         = -1

```

```

885     , before-sep      = 4ex
886     , after-sep       = 3ex
887     , number-format   = \partname\nobreakspace\thepart
888     , decls           = \raggedright\bfseries
889     , number-decls    = \Large
890     , title-decls     = \huge
891     , headformat-instance = part
892     , mark-cmd        = \partmark {#1}
893   }
894   \DeclareInstance{headformat}{part}{display}
895   {
896     , indent          = 0pt
897     , before-code     =
898     , after-code      =
899     , number-title-sep = 0pt
900   }
901   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
902   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
903   \__head_setup_default_heading:
904 }
905
906 \AddToHook{class/report/after}[head/example]
907 {
908   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
909   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
910   \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
911   {
912     \if@mainmatter
913       \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
914     \else
915       \IfBooleanTF{#1}
916       {% starred:
917         \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
918       }
919       {\IfNoValueTF{#2}
920        {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {\shorttitle={#3}} {#3}}
921        {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}}
922     }
923     \fi
924   }
925   \__head_setup_default_heading:
926 }
927 \AddToHook{class/book/after}[head/example]
928 {
929   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
930   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
931   \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
932   {
933     \if@mainmatter
934       \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
935     \else
936       \IfBooleanTF{#1}
937       {% starred:
938         \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}

```

```

939     }
940     {\IfNoValueTF{#2}
941      {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {shorttitle={#3}} {#3}}
942      {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}
943     }
944     \fi
945   }
946   \__head_setup_default_heading:
947 }

948 \cs_new_protected:Npn \__head_setup_default_heading:
949 {

\section

950   \DeclareDocumentCommand \section {s = {shorttitle} o m}
951   { \ParseLaTeXeHeading {section} {##1} {##2} {##3} }
(End of definition for \section. This function is documented on page ??.)

\subsection

952   \DeclareDocumentCommand \subsection {s = {shorttitle} o m}
953   { \ParseLaTeXeHeading {subsection} {##1} {##2} {##3} }
(End of definition for \subsection. This function is documented on page ??.)

\subsubsection

954   \DeclareDocumentCommand \subsubsection {s = {shorttitle} o m}
955   { \ParseLaTeXeHeading {subsubsection} {##1} {##2} {##3} }
(End of definition for \subsubsection. This function is documented on page ??.)

\paragraph

956   \DeclareDocumentCommand \paragraph {s = {shorttitle} o m}
957   { \ParseLaTeXeHeading {paragraph} {##1} {##2} {##3} }
(End of definition for \paragraph. This function is documented on page ??.)

\subparagraph

958   \DeclareDocumentCommand \subparagraph {s = {shorttitle} o m}
959   { \ParseLaTeXeHeading {subparagraph} {##1} {##2} {##3} }
960 } %end of command
(End of definition for \subparagraph. This function is documented on page ??.)

961 \</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  $\text{\TeX}$  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.

```
962 <*\latex-lab>  
963 \ProvidesFile{sec-template-latex-lab-testphase.ltx}  
964 [\l_tlabsecIIdate\space v\l_tlabsecIIversion\space latex-lab wrapper sec-  
  template]  
965 \RequirePackage{latex-lab-testphase-sec-template}  
966 </\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.

Symbols	
<code>\@startsection</code> .....	<i>15</i>
A	
<code>\addcontents</code> .....	<i>11</i>
<code>\addcontentsline</code> .....	<i>16, 35, 666</i>
<code>\addcontentslinebookmark</code> .....	<i>628, 654</i>
<code>\addcontentslinebookmarkOff</code> ..	<i>628, 665</i>
<code>\addcontentslinebookmarkOn</code> .....	<i>628</i>
<code>\addcontentslinebookmarkReset</code> .....	<i>630, 638, 640, 674</i>
<code>\addpenalty</code> .....	<i>616</i>
<code>\addtocontents</code> .....	<i>707, 708</i>
<code>\AddToHook</code> <i>66, 325, 632, 833, 879, 906, 927</i>	
<code>\addvspace</code> .....	<i>617, 707, 708</i>
B	
<code>\begingroup</code> .....	<i>419</i>
<code>\bfseries</code> .....	<i>687, 702, 717, 727, 736, 745, 757, 888</i>
bool commands:	
<code>\bool_gset_false:N</code> .....	<i>18</i>
<code>\bool_gset_true:N</code> .....	<i>13</i>
<code>\bool_if:NTF</code> .....	<i>24, 26, 96, 449, 497, 554, 595, 656, 668</i>
<code>\bool_if_p:N</code> .....	<i>593</i>
<code>\bool_lazy_or_p:nn</code> .....	<i>591</i>
<code>\bool_new:N</code> .....	<i>8, 58</i>
<code>\bool_set:Nn</code> .....	<i>590</i>
<code>\bool_set_false:N</code> .....	<i>84</i>
<code>\bool_set_true:N</code> .....	<i>78</i>
<code>\BooleanFalse</code> .....	<i>9, 96</i>
<code>\BooleanTrue</code> .....	<i>9, 96, 917, 920, 921, 938, 941, 942</i>
C	
<code>\centering</code> .....	<i>687</i>
<code>\chapter</code> .....	<i>4, 7, 14, 15, 35, 36, 40, 861, 862, 864, 910, 931</i>
<code>\chaptermark</code> .....	<i>10, 706</i>
<code>\cleardoublepage</code> .....	<i>246, 263</i>
<code>\clearpage</code> .....	<i>10, 34, 248, 263, 377</i>
<code>\clubpenalty</code> .....	<i>417, 426</i>
<code>\cs</code> .....	<i>319</i>
cs commands:	
<code>\cs_gset_protected:Npx</code> .....	<i>23, 25</i>
<code>\cs_new:Npn</code> .....	<i>31, 39, 124, 139, 585, 589, 608, 645</i>
<code>\cs_new_eq:NN</code> .....	<i>9, 10</i>
<code>\cs_new_protected:Npn</code> .....	<i>11, 16, 21, 28, 29, 67, 948</i>
<code>\cs_to_str:N</code> .....	<i>837</i>
<code>\csname</code> .....	<i>328</i>
D	
<code>\DebugHeadingsOff</code> .....	<i>16, 28</i>
<code>\DebugHeadingsOn</code> .....	<i>16, 28</i>
<code>\DeclareDocumentCommand</code> .....	<i>791, 850, 864, 901, 908, 910, 929, 931, 950, 952, 954, 956, 958</i>
<code>\DeclareInstance</code> ...	<i>43, 680, 693, 710, 719, 729, 738, 750, 762, 768, 774, 780, 786, 804, 814, 818, 827, 881, 894</i>
<code>\DeclareTemplateCode</code> .....	<i>23, 198, 331, 434, 482, 540</i>
<code>\DeclareTemplateCopy</code> .....	<i>23, 176</i>
<code>\DeclareTemplateInterface</code> .....	<i>150, 177, 184, 191</i>
<code>\def</code> .....	<i>396</i>
dim commands:	
<code>\dim_compare:nNnTF</code> .....	<i>451, 461, 501, 514, 556, 566, 618, 624</i>
<code>\DocumentMetadata</code> .....	<i>2</i>
E	
<code>\EditTemplateDefaults</code> .....	<i>23</i>
<code>\else</code> .....	<i>247, 254, 263, 327, 425, 615, 639, 810, 815, 914, 935</i>
<code>\endcsname</code> .....	<i>326, 328</i>
<code>\endgroup</code> .....	<i>421</i>
<code>\everypar</code> .....	<i>413, 427, 614</i>
exp commands:	
<code>\exp_not:N</code> .....	<i>94, 807, 821</i>
<code>\exp_not:n</code> <i>33, 34, 35, 36, 37, 41, 42, 43, 44, 45, 46, 47, 48, 49, 70, 71, 95, 97, 98, 99, 100, 101, 102, 103, 104, 137, 304, 305, 306, 307, 308, 401, 402, 403, 404, 405, 677, 811, 824, 845</i>	
<code>\expandafter</code> .....	<i>328</i>
<code>\ExpandArgs</code> .....	<i>845, 859</i>
F	
<code>\fbox</code> .....	<i>4</i>
<code>\fi</code> <i>249, 256, 263, 284, 285, 288, 329, 428, 611, 620, 641, 800, 810, 828, 923, 944</i>	

**G**

\global ..... 265, 412, 415

group commands:

  \group\_begin: ..... 444, 492, 550

  \group\_end: ..... 480, 538, 583

**H**

head commands:

  \head\_debug\_off: ..... 16, 11, 16, 29

  \head\_debug\_on: ..... 16, 11, 11, 28

head internal commands:

  \l\_head\_after\_code\_tl .....  
   ..... 438, 477, 486, 536, 544, 582

  \l\_head\_after\_penalty\_skip ....  
   ..... 220, 354, 618, 619, 624, 625

  \l\_head\_after\_skip 221, 318, 355, 424

  \l\_head\_before\_code\_tl .....  
   ..... 437, 475, 485, 534, 543, 580

  \l\_head\_before\_skip ... 218, 352, 617

  \l\_head\_bookmark\_tl .....  
   ..... 52, 77, 83, 100, 110

  \l\_head\_contents\_extra\_tl ....  
   ..... 231, 365, 676

  \\_\_head\_debug:n ..... 9, 9, 23

  \g\_head\_debug\_bool .. 8, 13, 18, 24, 26

  \\_\_head\_debug\_gset: .... 11, 14, 19, 21

  \\_\_head\_debug\_typeout:n ..... 9,  
   10, 25, 32, 33, 34, 35, 36, 37, 40,  
   41, 42, 43, 44, 45, 46, 47, 48, 49,  
   68, 69, 92, 127, 132, 137, 143, 206,  
   208, 210, 300, 342, 344, 397, 677, 793

  \l\_head\_decls\_tl .....  
   ..... 225, 359, 448, 496, 553

  \\_\_head\_determine\_number\_  
   typesetting:N ... 295, 390, 589, 589

  \\_\_head\_determine\_penalty: ....  
   ..... 294, 389, 585, 585

  \l\_head\_final\_code\_tl .....  
   ..... 24, 25, 28, 223, 272,  
   276, 292, 321, 357, 382, 385, 387, 430

  \\_\_head\_find\_label:w .... 72, 124, 124

  \\_\_head\_find\_label\_aux:w .....  
   ..... 21, 135, 139, 145

  \\_\_head\_handle\_marks\_etc:nnnnn .  
   ..... 313, 407, 645, 645

  \l\_head\_headformat\_instance\_tl  
   ..... 224, 301, 303, 358, 398, 400

  \l\_head\_indent\_dim .....  
   .... 436, 451, 453, 457, 461, 463,  
   467, 484, 501, 503, 507, 514, 516,  
   519, 542, 556, 558, 562, 566, 568, 571

  \l\_head\_instance\_keys\_tl .....  
   ..... 20, 59, 90, 91, 95

  \l\_head\_label\_tl .....  
   ... 19, 21, 59, 102, 122, 128, 133, 144

  \l\_head\_level\_int .....  
   ..... 201, 297, 299, 334,  
   392, 394, 395, 445, 493, 526, 575, 592

  \\_\_head\_mark\_cmd:n .... 213, 347, 649

  \l\_head\_name\_tl .....  
   . 200, 234, 333, 371, 454, 457, 464,  
   467, 503, 506, 516, 519, 559, 562,  
   568, 571, 601, 604, 654, 658, 666, 670

  \l\_head\_nameref\_tl .. 55, 86, 101, 113

  \l\_head\_number\_decls\_tl .....  
   ..... 226, 360, 469, 521, 574

  \\_\_head\_number\_format:n 230, 364, 604

  \l\_head\_number\_title\_sep\_tl ...  
   ..... 439, 472, 487, 522, 545, 577

  \l\_head\_penalty\_int .....  
   ..... 219, 353, 586, 587, 616

  \l\_head\_placement\_tl 59, 207, 209,  
   211, 240, 273, 341, 343, 345, 375, 383

  \l\_head\_pname\_tl ..... 203, 336

  \\_\_head\_process\_shorttitle:nN ..  
   ..... 844, 858, 877

  \l\_head\_quote\_decls\_tl ... 229, 363

  \l\_head\_quote\_tl ... 57, 88, 104, 121

  \l\_head\_reset\_cnt\_tl ..... 204, 337

  \l\_head\_running\_tl 53, 76, 82, 99, 111

  \l\_head\_saved\_secnumdepth\_tl ..  
   ..... 59, 235, 316, 370, 408

  \\_\_head\_setup\_default\_heading: .  
   ..... 903, 925, 946, 948

  \\_\_head\_show\_arguments:nnnnn ...  
   ..... 31, 442, 490, 548

  \\_\_head\_show\_arguments:nnnnnn ... 31

  \\_\_head\_show\_arguments:nnnnnnnnn  
   ..... 39, 39, 236, 368

  \l\_head\_start\_code\_tl .....  
   .... 24–26, 28, 34, 222, 239, 243,  
   261, 269, 356, 373, 377, 379, 609, 623

  \l\_head\_subtitle\_decls\_tl 228, 362

  \l\_head\_subtitle\_tl . 56, 87, 103, 120

  \l\_head\_title\_decls\_tl .....  
   ..... 227, 361, 474, 533, 579

  \l\_head\_title\_tl ..... 19, 21,  
   59, 81, 82, 83, 86, 97, 129, 134, 137, 140

  \l\_head\_tmpa\_tl ..... 51,  
   499, 512, 526, 530, 844, 845, 858, 859

  \l\_head\_toc\_tl ... 54, 75, 81, 98, 112

  \l\_head\_typeset\_number\_tl .....  
   ..... 33, 59, 305, 402, 598, 602

  \l\_head\_unnumbered\_bool .....  
   ..... 78, 84, 96, 115, 117,  
   449, 497, 526, 554, 590, 595, 656, 668

\_head\_vertical\_before\_spacing: ..... 296, 391, 608, 608	\_int\_set:Nn ..... 587
headformat (type) ..... 149	\_int\_use:N ..... 235, 297, 299, 370, 392, 394, 395, 445, 493, 526, 575
headformat display (instance) ..... 762	\_c\_max\_int ..... 22, 162, 586
headformat display (template) .. 177, 434	\_interlinepenalty ..... 32, 447, 495, 552
headformat hang (instance) ..... 768	
headformat hang (template) .... 184, 482	<b>K</b>
headformat paragraph (instance) .... 774	keys commands:
headformat runin (template) ... 191, 540	\_keys\_define:nn ..... 107
headformat section-special (instance) 786	\_keys\_set\_known:nnN ..... 91
headformat subparagraph (instance) .. 780	
heading (type) ..... 148	<b>L</b>
heading chapter (instance) ..... 693	\_label ..... 3, 8, 9, 19–21, 72, 122, 124, 133, 139, 144
heading display (template) .... 150, 198	\_Large ..... 717, 889
heading paragraph (instance) ..... 738	\_large ..... 727
heading part (instance) ..... 680	\_lastbox ..... 416
heading runin (template) ..... 176, 331	\_leavevmode ..... 611
heading section (instance) ..... 710	\_let ..... 638, 640
heading subparagraph (instance) .... 750	\_ltlabsecIIdate ..... 5, 964
heading subsection (instance) ..... 719	\_ltlabsecIIversion ..... 6, 964
heading subsubsection (instance) .... 729	
\\Huge ..... 689, 704	<b>M</b>
\\huge ..... 688, 703, 890	\\MakeLinkTarget 15, 454, 457, 464, 467, 503, 506, 516, 519, 559, 562, 568, 571
<b>I</b>	\\MakeUppercase ..... 12, 40
\\if... ..... 44	\\markboth ..... 66
\\IfBlankF ..... 648, 652	\\MessageBreak ..... 847, 848, 861, 862, 870, 871, 873
\\IfBlankT ..... 665	
\\IfBlankTF ..... 650	<b>N</b>
\\IfBooleanT ..... 69	\\nameref ..... 3
\\IfBooleanTF ..... 73, 874, 915, 936	\\newcommand ..... 628, 629, 630
\\ifcsname ..... 326	\\NewCommandCopy ..... 832
\\ifdim ..... 797, 802	\\NewDocumentCommand ..... 9, 877
\\IfInstanceExistsF ..... 792	\\newif ..... 27, 44, 328
\\IfNoValueTF ..... 89, 841, 855, 919, 940	\\newpage ..... 278, 283
\\IfValueT ..... 70	\\NewTemplateType ..... 148, 149
\\ignorespaces ..... 323, 432	\\nobreak ..... 317, 471
instances:	\\nobreakspace ..... 686, 887
headformat display ..... 762	\\noexpand .... 847, 848, 861, 862, 870, 872
headformat hang ..... 768	\\NoValue ..... 8
headformat paragraph ..... 774	\\normalcolor ..... 446, 494, 551
headformat section-special .... 786	\\normalfont ..... 11, 12, 167, 446, 494, 551, 717, 727, 736, 745, 757
headformat subparagraph ..... 780	\\normalsize ..... 736, 745, 757
heading chapter ..... 693	\\NoValue ..... 9
heading paragraph ..... 738	\\null ..... 257, 281
heading part ..... 680	\\numberline ..... 658, 670
heading section ..... 710	
heading subparagraph ..... 750	<b>O</b>
heading subsection ..... 719	\\onecolumn ..... 252
heading subsubsection ..... 729	\\openright ..... 44
int commands:	
\\int\_compare:nNnTF ..... 586	
\\int\_compare\_p:nNn ..... 592	
\\int\_gset:Nn ..... 316, 408, 597	

<b>P</b>	
<code>\par</code> .....	317, 471, 478, 537, 612
<code>\paragraph</code> .....	13, 14, 956
<code>\paragraphmark</code> .....	746
<code>\parindent</code> .....	702, 782
<code>\ParseLaTeXeHeading</code> .....	7, 9, 19, 20, 36, 67, 830, 842, 845, 851, 856, 859, 865, 902, 909, 913, 917, 920, 921, 930, 934, 938, 941, 942, 951, 953, 955, 957, 959
<code>\part</code> .....	3, 4, 7, 14, 15, 18, 25, 36, 40, 847, 848, 850, 901, 908, 929
<code>\partmark</code> .....	3
<code>\partmark</code> .....	66, 691, 892
<code>\partname</code> .....	686, 887
<code>\protect</code> .....	658, 670, 707, 708
<code>\providecommand</code> .....	66, 631
<code>\ProvidesExplPackage</code> .....	3
<code>\ProvidesFile</code> .....	963
<b>Q</b>	
quark commands:	
<code>\q_no_value</code> .....	21, 72
<code>\quark_if_no_value:nTF</code> ....	125, 141
<b>R</b>	
<code>\raggedright</code> .....	702, 888
<code>\refstepcounter</code> .....	15, 33
<code>\relax</code> .....	640, 795, 801, 842, 845
<code>\renewcommand</code> .....	635
<code>\RenewCommandCopy</code> .....	634, 834
<code>\RenewDocumentCommand</code> .....	835
<code>\RequirePackage</code> .....	965
<b>S</b>	
<code>\secdef</code> .....	15
<code>\secdef</code> .....	4, 7, 15, 36, 40, 835
<code>\section</code> .....	5, 7, 8, 13, 14, 36, 39, 950
<code>\sectionmark</code> .....	10, 714
<code>\setbox</code> .....	416
<code>\SetCurrentTemplateKeys</code> .....	25, 44
<code>\SetTemplateKey</code> .....	25
<code>\SetTemplateKeys</code> ..	238, 372, 443, 491, 549
skip commands:	
<code>\skip_horizontal:N</code> .....	424, 453, 457, 463, 467, 503, 507, 516, 519, 558, 562, 568, 571
<code>\skip_horizontal:n</code> .....	522, 577
<code>\skip_vertical:N</code> .....	318
<code>\skip_vertical:n</code> .....	472
<code>\c_zero_skip</code> ..	451, 461, 501, 514, 556, 566
<code>\space</code> .....	701, 964
<code>\specialsection</code> .....	39
str commands:	
<code>\str_case:nn</code> .....	375
<code>\str_case:nnTF</code> ....	240, 273, 383, 837
<code>\string</code> .....	794
<code>\subparagraph</code> .....	13, 14, 958
<code>\subparagraphmark</code> .....	758
<code>\subsection</code> .....	13, 14, 952
<code>\subsectionmark</code> .....	724
<code>\subsubsection</code> .....	13, 14, 954
<b>T</b>	
tag commands:	
<code>\tag_if_active:TF</code> .....	869
template types:	
<code>headformat</code> .....	149
<code>heading</code> .....	148
templates:	
<code>headformat display</code> .....	177, 434
<code>headformat hang</code> .....	184, 482
<code>headformat runin</code> .....	191, 540
<code>heading display</code> .....	150, 198
<code>heading runin</code> .....	176, 331
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\OM</code> .....	417, 447, 495, 552
<code>\@afterheading</code> .....	292
<code>\@afterindentfalse</code> .....	216, 799
<code>\@afterindenttrue</code> .....	215, 796
<code>\@chapapp</code> .....	701
<code>\@chapter</code> .....	4
<code>\@clubpenalty</code> .....	426
<code>\@currentlabelname</code> .....	647
<code>\@firstoftwo</code> .....	9
<code>\@hangfrom</code> .....	528
<code>\@kernel@refstepcounter</code> .....	601
<code>\@latex@error</code> .....	869
<code>\@latex@warning</code> .....	847, 861, 869
<code>\@minus</code> .....	743, 755
<code>\@nobreakfalse</code> .....	411
<code>\@noskipsecfalse</code> .....	415
<code>\@noskipsectrue</code> .....	412
<code>\@part</code> .....	4
<code>\@plus</code> .....	743, 755
<code>\@secCNTformat</code> .....	9
<code>\@secondoftwo</code> .....	9
<code>\@secpenalty</code> .....	11, 12, 33, 587
<code>\@startsection</code> .....	4, 5, 7, 10, 15, 36, 39, 40, 43, 791, 832, 834
<code>\@svsechd</code> .....	396, 420
<code>\@tempskipa</code> ....	795, 797, 798, 808, 822
<code>\@tempskipb</code> ....	801, 802, 809, 816, 823
<code>\@tempswa</code> .....	25
<code>\@tempswafalse</code> .....	255
<code>\@tempswatrue</code> .....	253
<code>\@topnum</code> .....	265
<code>\c@secnumdepth</code> .....	33, 235, 316, 370, 408, 592, 597



