

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

$\text{\LaTeX}$ Project*

v0.9o 2026-09-14

Abstract

Contents

1	Introduction	3
2	Changes to the user interface	3
2.1	New user commands	3
2.2	The optional argument	3
2.3	Class support	4
2.4	Changing heading commands	5
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	7
2.5.3	Step 3: Declare the instances	8
2.5.4	Step 4 Redefine the <code>\section</code> command	8
3	Setting up $\text{\LaTeX}$ 2ϵ heading commands in classes	8
4	Open points	9
5	Template types and templates for headings	9
5.1	Template types	9
5.1.1	The template type ‘heading’	9
5.1.2	The template type ‘headformat’	10
5.2	Templates	11
5.2.1	Templates of type <code>heading</code>	11
5.2.2	The <code>heading</code> template ‘ <code>\all</code> ’	11
5.2.3	The <code>heading</code> template ‘display’	12
5.2.4	The <code>heading</code> template ‘runin’	14
5.2.5	Templates of type <code>headformat</code>	15
5.2.6	The <code>headformat</code> template ‘display’	16
5.2.7	The <code>headformat</code> template ‘hang’	16
5.2.8	The <code>headformat</code> template ‘runin’	17
5.3	Default instances	17

*Initial implementation by Frank Mittelbach.

5.3.1	Instances of <code>heading</code> templates	17
5.3.2	Instances of <code>headformat</code> templates	18
5.4	Use of templates from the 2026-06-01 release of <code>L^AT_EX</code>	18
6	Support for legacy classes and packages	19
6.1	Support classes based on legacy <code>L^AT_EX 2_ε</code> interfaces	19
6.2	Support for the <code>titlesec</code> package interfaces	19
7	Support for links	20
8	Bookmarks	20
9	Tagging support	20
10	Debugging support	20
11	The Implementation	21
11.1	Debugging	21
11.2	Temp variable(s)	22
11.3	variable(s)	22
11.4	New user commands	23
11.5	The <code>L^AT_EX 2_ε</code> parsing interface	23
11.6	General helper commands	26
11.7	Templates	27
11.7.1	Template types	27
11.7.2	<code>heading</code> templates interfaces	28
11.7.3	<code>headformat</code> templates interfaces	30
11.7.4	<code>heading</code> templates code	31
11.7.5	<code>headformat</code> templates code	41
11.7.6	Internal commands used by the template code	45
11.8	Support for legacy classes and packages	48
11.8.1	Instances (sample/default definitions)	49
11.8.2	New <code>\@startsection</code>	52
11.8.3	New <code>\secdef</code>	55
11.8.4	Core <code>L^AT_EX</code>	56
12	Issues and problems noticed along the way	58
12.1	Local <code>\DeclareInstance</code>	58
12.2	<code>\SetCurrentTemplateKeys</code>	58
12.3	O-expansion of <code>\UseInstance</code> arguments	58
12.4	Switch <code>\openright</code> is not defined by core	58
12.5	Indexheading at least for <code>l3doc</code> is wrong now	58

A	Templates from the 2026-06-01 release of L^AT_EX	58
A.1	Template documentation for -v1 templates	59
A.1.1	Templates of type <code>heading</code>	59
A.1.2	The <code>heading</code> template ‘ <code><all></code> ’	59
A.1.3	The <code>heading</code> template ‘ <code>display-v1</code> ’	60
A.1.4	The <code>heading</code> template ‘ <code>runin-v1</code> ’	61
A.1.5	Templates of type <code>headformat</code>	62
A.1.6	The <code>headformat</code> template ‘ <code>hang-v1</code> ’	62
A.1.7	The <code>headformat</code> template ‘ <code>runin-v1</code> ’	63
A.1.8	The <code>headformat</code> template ‘ <code>display-v1</code> ’	63
A.2	Implementation of -v1 templates	64
	Index	73

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

<code>\theheading</code>	This command expands inside a heading to the current number.
<code>\partmark</code>	This replaces the fix <code>\markboth{}{}</code> or similar in the standard <code>\part</code> 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 key `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 running 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 an empty value suppresses the bookmark entry.

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

label This sets a label, so it 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^AT_EX heading commands.

shorttitle This 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. If **shorttitle** is used together with any of the individual keys (**toc**, **running**, **bookmark**, **nameref**) then they overwrite the default value provided by **shorttitle**.

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 commands `\@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.

Heading commands in other classes defined with `\secdef` will likely error as the code has no real chance to know how to handle such a heading.

Heading commands which use neither `\@startsection` nor `\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.

For the AMS-classes, `latex-lab-firstaid` contains instances for `\part`, `\chapter`, `\section`, and all other headings used by these classes. The current set of instances is now assumed to reproduce the original layout. Setting them up revealed a number of interesting differences between these classes—not all of them intentional (I think).

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 (here `section`) 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 or by declaring (instead of editing) an instance with this name in the preamble. 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 restriction 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
> force-unnumbered => false
> placement => normal
> column-spanning => false
> mark-cmd => \sectionmark {##1}
> para-indent => false
> before-vspace => 8.0pt plus 2.0pt minus 2.0pt
> penalty => \c_max_int
> after-penalty-vspace => 0pt
> after-vspace => 4.0pt
> start-code =>
> final-code =>
> prefix => \NoValue
> punct =>
> heading-decls => \tubsecfmt
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> heading-format => ##1
> prefix-format => ##1
> number-format => ##1
> punct-format => ##1
> title-format => \use:n {##1} \par
> subtitle-format => ##1
> quote-format => ##1
> number-tag => heading-number
> headformat-instance => section-@startsection
> contents-extra =>
> name => section
> from template => display.
```

The instance 'section-@startsection' of type 'headformat' has values:

```
> heading-indent => Opt
> order-hang => prefix,separator-a,?,number,punct,separator-b,?
> order => title,separator-c,subtitle
> separator-a => \nobreakspace
> separator-b => \hspace {1em}
> separator-c => \
> separator-d =>
> 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
> force-unnumbered => false
> placement => normal
> column-spanning => false
> mark-cmd =>
> para-indent => false
> before-vspace => Opt
> penalty => \c_max_int
> after-penalty-vspace => Opt
> after-vspace => Opt
> start-code =>
> final-code =>
> prefix => \NoValue
> punct =>
> heading-decls => \normalfont
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> heading-format => ##1
> prefix-format => ##1
> number-format => ##1
```

```

> punct-format => ##1
> title-format => ##1\par
> subtitle-format => ##1
> quote-format => ##1
> number-tag => heading-number
> headformat-instance => std
> contents-extra => .

```

The template 'hang' of type 'headformat' has default values:

```

> heading-indent => 0pt
> order-hang => prefix,separator-a,?,number,punct,separator-b,?
> order => title,separator-c,subtitle
> separator-a => \nobreakspace
> separator-b => \hspace {1em}
> separator-c => \
> separator-d => .

```

2.5.3 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}
{
  level          = 1,
  mark-cmd       = \sectionmark{#1}, % remove second hash
  before-vspace  = 8.0pt plus 2.0pt minus 2.0pt,
  after-vspace   = 4.0pt,
  heading-decls  = \tubsecfmt,
  headformat-instance = section,      % new name
}

```

The value for key `title-format` seems to have changed as well but `\use:n {##1} \par` is really only a fancy way to write `##1 \par` so we can keep the default. Note, however, that the `##1` have to be replaced by `#1` when you reenter them.

In case of the `headformat` instance only default values have been used so all that is necessary is:

```

\DeclareInstance{headformat}{section}{hang}{}

```

Of course, if you do not want to rely on default values you can (for clarity) specify all keys and their values in these instances.

2.5.4 Step 4 Redefine the `\section` command

Finally, the heading command should be defined to use the new interface:

```

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

```

What `\ParseLaTeXeHeading` does is explained in the next section.

3 Setting up L^AT_EX 2_ε heading commands in classes

Heading commands are managed by defining an instance of a `heading` template which is then used in `\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 adjust the layout of the heading. Examples of instances that mimic the behavior of the heading commands in the standard classes can be found below. Section 2.5 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 automatically 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 6.2.

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 used 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. This includes doing all the work necessary for tagging.

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 then `\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 fixed prefix text

Arg: 3 formatted heading number

Arg: 4 main title of the heading

Arg: 5 subtitle

Arg: 6 quotation

Semantics:

Handles the layout of just the heading label (prefix and number), 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 prefix, number, subtitle or quotation then this is indicated with `\NoValue`.

The templates are currently implemented to mimic L^AT_EX 2_ε behavior so if a `heading` template should not number a heading it calls a `headformat` template with both prefix and number set to `\NoValue`.

The `<subtitle>` and `<quotation>` are always ignored by the currently defined templates but that will probably change soon, either by extending them or by providing additional ones that support these mandatory argument.

5.2 Templates

5.2.1 Templates of type heading

There are quite 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., `heading-decls`) are all declared in the `heading` templates even if they are actually only used within `headformat` templates. In other words, `headformat` templates have a number of implicit variables that they expect to be set.¹

5.2.2 The heading template ‘<all>’

Attributes:

name (*tokenlist*) Referenceable 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 — not yet implemented/may vanish

reset-counter (*tokenlist*) Name of the heading instance that should reset the numbering of this heading level (if any) — not yet implemented/may vanish

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`

force-unnumbered (*boolean*) This heading is never numbered, even if explicitly requested in the optional argument to the heading. If **false** then numbering is determined in the normal way (through `secnumdepth`, `*`, or a setting in the optional argument to the heading).

Default: **false**

¹Maybe questionable

- 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), **ragged** (like **normal** but if a page break is taken before the heading then the previous page is set ragged and not stretched). 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.
- column-spanning** (*boolean*) If true produces a heading that spans columns in **twocolumn** typesetting. Requires **placement = top** (and adjusts if necessary) and only has an effect if the columns are produced via the **twocolumn** class option and not when using **multicol**. This might get extended when we refactor the OR handling. Default: **false**
- prefix** (*tokenlist*) A fixed string, such as “Chapter”, that can be used together with the number (if any) to form a “heading label”. Usage and placement is up to the template, so it could be placed after the number by the template (in which case it isn’t really a prefix). Default: **\NoValue**
- punct** (*tokenlist*) A fixed string to be added to the end of the heading number, i.e., it shows up in on the heading but not in cross-references. Default: **<empty>**
- 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^AT_EX 2_ε 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>`.

5.2.3 The heading template ‘display’

Attributes:

- para-indent** (*boolean*) Should the paragraph after the heading be indented? Default: **false**
- before-vspace** (*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: **Opt**
- penalty** (*integer*) Penalty to break before the heading. The default (T_EX’s largest integer) indicates that no penalty was set in which case `\@secpenalty` is used (to support the L^AT_EX 2_ε interface). Default: **1073741823**

after-penalty-vspace (*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-vspace (*skip*) Vertical space after the heading Default: 0pt

heading-decls (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Note that color commands (unless `luacolor` is used) introduce a break point before the heading and therefore one should either surround color commands with `\SaveLastSkip` and `\RestoreLastSkip` or to use the keys `prefix-decls`, `number-decls`, `title-decls`, etc. instead. Default: `\normalfont`

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

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

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

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

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

heading-format (*function(1)*) Code that receives the whole heading as an argument. Default: #1

prefix-format (*function(1)*) Code that receives the prefix string as an argument. Default: #1

number-format (*function(1)*) Code that receives the number string as an argument. Default: #1

punct-format (*function(1)*) Code that receives the punctuation string as an argument. Default: #1

title-format (*function(1)*) Code that receives the title string as an argument. By default it ends in `\par` so that `\baselineskip` changes in the settings (e.g., in `title-decls`) are applied Default: #1 `\par`

subtitle-format (*function(1)*) Code that receives the subtitle string as an argument. Default: #1

quote-format (*function(1)*) Code that receives the quote string as an argument. Default: #1

number-tag (*tokenlist*) Name of the tag to put around the number (empty means no tag). Default: `heading-number`

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

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

Semantics & Comments: The key **heading-decls** determines the overall `\baselineskip` used in the heading. Font changes in individual declarations do not have that effect, because they are all executed in groups and if all elements are within the same paragraph then the final `\par` after the heading comes too late. If you want that the font setting for a specific key, e.g., **title**, depends on the font setting in **title-decls** you have to ensure that a `\par` happens within **title-format** key, e.g., `title-format=#1\par` (which is what happens in the default value of **title-format**).

5.2.4 The heading template ‘runin’

Attributes:

before-vspace (*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 (T_EX’s largest integer) indicates that no penalty was set in which case `\@secpenalty` is used. Default: **1073741823**

after-penalty-vspace (*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-hspace (*skip*) Horizontal space after the heading. Default: **0pt**

heading-decls (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Note that color commands (unless **luacolor** is used) introduce a break point before the heading and therefore one should either surround color commands with `\SaveLastSkip` and `\RestoreLastSkip` or to use the keys **prefix-decls**, **number-decls**, **title-decls**, etc. instead. Default: **\normalfont**

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

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

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

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

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

heading-format (*function(1)*) Code that receives the whole heading as an argument.
Default: **#1**

prefix-format (*function(1)*) Code that receives the prefix string as an argument.
Default: **#1**

number-format (*function(1)*) Code that receives the number string as an argument.
Default: **#1**

punct-format (*function(1)*) Code that receives the punctuation string as an argument.
Default: **#1**

title-format (*function(1)*) Code that receives the title string as an argument. Since this is a runin heading we should remain in horizontal mode so the default does not include a `\par`.
Default: **#1**

subtitle-format (*function(1)*) Code that receives the subtitle string as an argument.
Default: **#1**

quote-format (*function(1)*) Code that receives the quote string as an argument.
Default: **#1**

number-tag (*tokenlist*) Name of the tag to put around the number (empty means no tag).
Default: **heading-number**

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

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

Semantics & Comments: The runin template is fairly similar to the display one. Main differences are due to the fact that we stay in horizontal mode at the end of the heading so that we have **after-hspace** instead of **after-vspace** and we do not have a `\par` inside **title-format**.

5.2.5 Templates of type **headformat**

The **headformat** templates deal with positioning prefix, number, punct, title, subtitle and quote. If the heading is unnumbered then prefix, number and punct are expected to be `\NoValue` (this is arranged in the calling **heading** template).

Tagging is taken care of in the calling template as well, all the the **headformat** templates do is opening and closing MCs as necessary.

The templates make use of the generic order key mechanism.

5.2.6 The headformat template ‘display’

Attributes:

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

order (*commalist*) Order of elements that form the heading. Some of the separators are dropped if the previous element is \NoValue, e.g., **separator-a** and **separator-b**. The keyquote key is currently ignored. Default: **prefix**, **separator-a**, **?**, **number**, **punct**, **separator-b**, **?**, **title**, **separator-c**, **subtitle**

separator-a (*tokenlist*) By default the separation between **prefix** and **number**. Default: \nobreakspace

separator-b (*tokenlist*) By default the separation between **punct** and **title**. The default starts a new line with extra vertical space. Default: \par \nobreak \vspace{20pt}

separator-c (*tokenlist*) By default the separation between **title** and **subtitle**. Default: \

separator-d (*tokenlist*) By default not used. Default: *<empty>*

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

This template can be called from the **heading display** template. It is not suitable for use with the **runin** template.

5.2.7 The headformat template ‘hang’

Attributes:

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

order-hang (*commalist*) Order of elements of the heading that are used to form the hanging indentation (and are placed in that space). Default: **prefix**, **separator-a**, **?**, **number**, **punct**, **separator-b**, **?**

order (*commalist*) Order of remaining headings elements. The keyquote key is currently not used. Default: **title**, **separator-c**, **subtitle**

separator-a (*tokenlist*) By default the separation between **prefix** and **number**. Default: \nobreakspace

separator-b (*tokenlist*) By default the horizontal separation after **punct** still being counted in the hanging indentation Default: \hspace{1em}

separator-c (*tokenlist*) By default the separation between **title** and **subtitle**. Default: \

separator-d (*tokenlist*) By default not used. Default: *<empty>*

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 L^AT_EX always used by default for `\section`, `\subsection`, and `\subsubsection`.

This template can be called from the `heading display` template. It is not suitable for use with the `runin` template.

5.2.8 The `headformat` template ‘`runin`’

Attributes:

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

order (*commalist*) Order of elements that form the heading. Some of the separators are dropped if the previous element is `\NoValue`, e.g., `separator-a` and `separator-b`. The keyquote key is currently ignored. Default: `prefix, separator-a, ?, number, punct, separator-b, ?, title, separator-c, subtitle`

separator-a (*tokenlist*) By default the separation between `prefix` and `number`. Default: `\nobreakspace`

separator-b (*tokenlist*) By default the separation between `punct` and `title`. The default starts a new line with extra vertical space. Default: `\hspace{1em}`

separator-c (*tokenlist*) By default the separation between `title` and `subtitle`. Default: `\`

separator-d (*tokenlist*) By default not used. Default: `\empty`

Semantics & Comments: Implements a heading layout where text after the heading continues on the same line as the heading. It is what L^AT_EX always used by default for `\paragraph` and `\subparagraph`.

This template is intended to be used with the `heading runin` template (but could perhaps also be used with `display`).

5.3 Default instances

These instances are set up to mimic the design of the standard L^AT_EX 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

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

chapter (instance of display template) Used in heading instance for `\chapter`. By default identical to the one used for `\part`.

std (instance of hang template) Used as default in the heading template if nothing else is specified.

section (instance of hang template) Used in heading instance for `\section`. By default identical to the `std` instance.

subsection (instance of hang template) Used in heading instance for `\subsection`. By default identical to the `std` instance.

subsubsection (instance of hang template) Used in heading instance for `\subsubsection`. By default identical to the `std` instance.

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

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

5.4 Use of templates from the 2026-06-01 release of L^AT_EX

In the first heading implementation using templates we provided a number of somewhat restricted templates that covered the default layouts of standard L^AT_EX classes but not much beyond this.

For the 2026-11-01 release we have replaced them with more general templates described above that use the general order key mechanism. As a side effect the new templates have some additional keys and some keys used in the templates of first implementation got dropped.

For the next couple of releases we will keep the original templates available (only renamed to `<name>-v1`). Eventually, we want to drop the `-v1` versions but for the next couple of releases we provide them alongside the new more flexible templates, so that there is no need to transitioning to the new templates in a rush.

If you have defined instances using the first template version then you can (for now) continue to use them simply by changing the template names in your instance declarations from `<name>` to `<name>-v1`, e.g., you have to change from

```
\DeclareInstance{heading}    {chapter}{display} { ... }
\DeclareInstance{headformat}{chapter}{display} { ... }

\DeclareInstance{heading}    {section}{hang} { ... }
\DeclareInstance{headformat}{section}{hang} { ... }
...
```

to

```
\DeclareInstance{heading}    {chapter}{display-v1} { ... }
\DeclareInstance{headformat}{chapter}{display-v1} { ... }

\DeclareInstance{heading}    {section}{hang-v1} { ... }
\DeclareInstance{headformat}{section}{hang-v1} { ... }
...
```

No other changes are necessary for now. However, going forward you should switch to the extended templates because the `-v1` templates will eventually be dropped.

Switching mainly means stopping the use of the keys `prefix-number-sep` and `number-title-sep` because they have been replaced with the more general `separator-a`, `separator-b`, ... keys.²

The template documentation for these deprecated templates is available as part of Appendix A in case you need to consult it in order to update your template instance declarations to use the new templates. Please do not use these templates for new work, as they will eventually get removed.

6 Support for legacy classes and packages

6.1 Support classes based on legacy L^AT_EX 2_ε 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 automatically declared unless they already exist.

`\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^AT_EX'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-vspace}{after-vspace}[right-sep]
```

²There are some more changes in the keys, but it is less likely that they have been used. If necessary, compare the documentation for the old and the new templates to find out what else needs adjusting.

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 no automatic target.

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

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 place 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`.

UFi:check if this is still the implementation ...

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 {\ltxlabsecIIdate}
6 {\ltxlabsecIIversion}
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:Npe \__head_debug:n ##1
24   { \bool_if:NT \g__head_debug_bool {##1} }
25   \cs_gset_protected:Npe \__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 20.)

\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 20.)

```
\_head_show_arguments:nnnnnn
31 \cs_new:Npn \_head_show_arguments:nnnnnn #1#2#3#4#5#6 {
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:~ prefix~ =~ \exp_not:n{#2}}
35   \_head_debug_typeout:n{3:~ number~ =~ \exp_not:n{#3}}
36   \_head_debug_typeout:n{4:~ title~ =~ \exp_not:n{#4}}
37   \_head_debug_typeout:n{5:~ subtitle~ =~ \exp_not:n{#5}}
38   \_head_debug_typeout:n{6:~ quotation~ =~ \exp_not:n{#6}}
39 }
```

(End of definition for `\_head_show_arguments:nnnnnn`.)

```
\_head_show_arguments:nnnnnnnnnn
Some debug data ...
40 \cs_new:Npn \_head_show_arguments:nnnnnnnnnn #1#2#3#4#5#6#7#8#9 {
41   \_head_debug_typeout:n{-----~ Heading~ instance~ arguments:}
42   \_head_debug_typeout:n{1:~ keys~ =~ \exp_not:n{#1}}
43   \_head_debug_typeout:n{2:~ unnumbered~ =~ \exp_not:n{#2}}
44   \_head_debug_typeout:n{3:~ title~ =~ \exp_not:n{#3}}
45   \_head_debug_typeout:n{4:~ toc~ =~ \exp_not:n{#4}}
46   \_head_debug_typeout:n{5:~ running~ =~ \exp_not:n{#5}}
47   \_head_debug_typeout:n{6:~ bookmark~ =~ \exp_not:n{#6}}
48   \_head_debug_typeout:n{7:~ nameref~ =~ \exp_not:n{#7}}
49   \_head_debug_typeout:n{8:~ label~ =~ \exp_not:n{#8}}
50   \_head_debug_typeout:n{9:~ subtitle | quote ~ =~ \exp_not:n{#9}}
51 }
```

(End of definition for `\_head_show_arguments:nnnnnnnnnn`.)

11.2 Temp variable(s)

```
\l__head_tmpa_tl
52 \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.

```
53 %\tl_new:N \l__head_bookmark_tl
54 %\tl_new:N \l__head_running_tl
55 %\tl_new:N \l__head_toc_tl
56 %\tl_new:N \l__head_nameref_tl
57 %\tl_new:N \l__head_subtitle_tl
58 %\tl_new:N \l__head_quote_tl
59 %\bool_new:N \l__head_unnumbered_bool
```

```
\l__head_label_tl
\l__head_instance_keys_tl
\l__head_number_tl
\l__head_saved_secnumdepth_tl
\l__head_title_tl
\l__head_placement_tl
60 \tl_new:N \l__head_label_tl
61 \tl_new:N \l__head_instance_keys_tl
```

But these token lists needs a declaration:

```

62 \tl_new:N \l__head_number_tl
63 \tl_new:N \l__head_saved_secnumdepth_tl
64 \tl_new:N \l__head_title_tl
65 \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.

```

66 \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.

```

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

```

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

11.5 The L^AT_EX 2_ε parsing interface

L^AT_EX 2_ε 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).

```

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

```

73   \__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^AT_EX 2_ε logic).

```

74 \IfBooleanTF{#2}
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 { \tl_set_eq:NN \l__head_toc_tl \l__head_title_tl
81   \tl_set_eq:NN \l__head_running_tl \l__head_title_tl
82   \tl_set_eq:NN \l__head_bookmark_tl \l__head_title_tl
83   \bool_set_false:N \l__head_unnumbered_bool
84 }

```

The nameref always starts out matching the main title.

```

85 \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`

```

86 \tl_set_eq:NN \l__head_subtitle_tl \c_novalue_tl
87 \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?

```

88 \IfNoValueTF { #3 }
89 { \tl_clear:N \l__head_instance_keys_tl }
90 { \keys_set_known:nN {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.³

```

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

```

³I think this is a common requirement and perhaps `\UseInstance` should really o-expand all arguments it receives.

```

103             { \exp_not:o \l__head_quote_tl } }
104     }
105 }

```

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.

```

106 \keys_define:nn {heading} {
107   , shorttitle .meta:n =
108     {toc = {#1} , running = {#1} , bookmark = {#1} , nameref= {#1} }
109   , bookmark .tl_set:N = \l__head_bookmark_tl
110   , running .tl_set:N = \l__head_running_tl
111   , toc .tl_set:N = \l__head_toc_tl
112   , nameref .tl_set:N = \l__head_nameref_tl
113   %
114   , numbered .bool_set_inverse:N = \l__head_unnumbered_bool
115   , numbered .default:n = true
116   , unnumbered .bool_set:N = \l__head_unnumbered_bool
117   , unnumbered .default:n = true
118   %
119   , subtitle .tl_set:N = \l__head_subtitle_tl
120   , 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.

```

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

```

(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.

```

123 \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.

```

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

```

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

```

129   { \__head_debug_typeout:n{---label~found~#2}
130     \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`.

```

131     \tl_set:Nn\l__head_title_tl {#1}
132     \__head_find_label_aux:w #3\__head_find_label_aux:w
133   }
134   \__head_debug_typeout:n{---return:~ '\exp_not:o \l__head_title_tl' }
135 }

```

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.

```
136 \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.

```
137 \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.

```
138 \quark_if_no_value:nF {#2}
139 { \__head_debug_typeout:n{---- extra~ label~ '#2'~ found }
140 \tl_put_right:Nn \l__head_label_tl { \label{#2} }
141 \__head_find_label_aux:w #3\__head_find_label_aux:w
142 }
143 }
```

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

11.6 General helper commands

`\WordSpaceAmount` Produce the amount of a (fractional) word space in the current font in a way that it can be used in a skip or dimen register assignment. The argument specifies the fraction, e.g., 2 gives two word spaces and .5 would produce half a word space.

This general helper doesn't really belong here, so should be eventually moved.

```
144 \newcommand\WordSpaceAmount[1]{
145 \glueexpr
146 #1\fontdimen2\the\font
147 plus #1\fontdimen3\the\font
148 minus #1\fontdimen4\the\font
149 \relax
150 }
```

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

Some designs automatically add a punctuation symbol (typically a period) at the end of a title, but that should only happen if the title doesn't already end in a punctuation symbol.

This is achieved by setting the `\spacefactor` for punctuation symbols (slightly) higher than 1000 and then checking the current `\spacefactor` before trying to add the punctuation. If it is 1000 or less then the title didn't end in a punctuation and we can safely add one, otherwise we don't.

If `\nonfrenchspacing` is in force this is automatically the case, but in $\text{\LaTeX 2}_{\epsilon}$ `\frenchspacing` did set the `\spacefactor` of all punctuation symbols to 1000 making them indistinguishable from other characters.

Thus, to make this work, we have to change `\frenchspacing` which is a breaking change as it can lead to pagination differences given that the word space after a punctuation gets (very minimally) larger this way. However, this approach was successfully used in the AMS classes for ages and it offers nice design possibilities so we enable it in documents using `\DocumentMetadata` automatically.

`\frenchspacing` We give all punctuation symbols a unique `\sfcode` value so that it is even possible to
`\nonfrenchspacing` determine which punctuation was just typeset.

```
151 \def\frenchspacing{\sfcode`\.\1006\sfcode`\?1005\sfcode`\!1004%
152 \sfcode`\:1003\sfcode`\;1002\sfcode`\,1001 }
```

The same should be done for `.?!` if we want to be able to distinguish those when `\nonfrenchspacing` is in force.

```
153 \def\nonfrenchspacing{\sfcode`\.\3002\sfcode`\?3001\sfcode`\!3000%
154 \sfcode`\:2000\sfcode`\;1500\sfcode`\,1250 }
```

(End of definition for `\frenchspacing` and `\nonfrenchspacing`. These functions are documented on page ??.)

`\nopunct` The AMS classes also offered `\nopunct` that could be added at the end of a title and would prevent the addition of a punctuation symbol.
 It should have a value for `\spacefactor` that is not used for `\frenchspacing` so that this special case can be detected.

```
155 \cs_new_protected:Npn \nopunct {\spacefactor 1007 }
```

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

`\AddPunct` In the AMS classes this was called `\@addpunct`.
 Note: this mechanism will fail if the text ending in a punctuation has an uppercase character just before the punctuation, e.g., if somebody writes `SOLUTION:`. In that case `SOLUTION\@.` would be necessary!

```
156 \cs_new_protected:Npn \AddPunct #1 {
157   \relax\ifhmode
158     \ifnum\spacefactor>\@m
159       \__head_debug_typeout:n {--->~ punctuation~ '#1'~ not~ added}
160     \else
161       \__head_debug_typeout:n {--->~ punctuation~ '#1'~ added}
162     #1
163   \fi
164 \fi
165 }
```

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

11.7 Templates

11.7.1 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 }

```
166 \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: prefix string
- 3: formatted heading number
- 4: title
- 5: subtitle
- 6: quote

```
167 \NewTemplateType{headformat}{6}
```

11.7.2 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.

```
168 \DeclareTemplateInterface{heading}{display}{9}
169 {
170   , name           : tokenlist
171   , parent-name    : tokenlist
172   , reset-counter  : tokenlist
173   , level          : integer = 0
```

By default headings can be numbered and the numbering is determined for each heading in the document individually.

```
174   , force-unnumbered : boolean = false
```

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

```
175   , placement       : choice {page , top , normal , ragged } = normal
176   , column-spanning : boolean = false
177   , mark-cmd        : function(1) =
178   , 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.

```
179   , before-vspace : skip = 0pt
180   , penalty       : integer = \c_max_int
181   , after-penalty-vspace : skip = 0pt
182   , after-vspace  : skip = 0pt
```

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

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

A prefix string such as `\@chappap` could be specified in the next variable. By default it has no value.

```
185   , prefix       : tokenlist = \NoValue
```

A postfix string to be added to the heading number (if the heading is numbered). By default empty (not \NoValue)!

```

186 , punct          : tokenlist =
187 , heading-decls   : tokenlist = \normalfont
188 , prefix-decls    : tokenlist =
189 , number-decls    : tokenlist =
190 , title-decls     : tokenlist =
191 , subtitle-decls  : tokenlist =
192 , quote-decls     : tokenlist =
193 , heading-format   : function{1} = #1
194 , prefix-format    : function{1} = #1
195 , number-format    : function{1} = #1
196 , punct-format     : function{1} = #1
197 , title-format     : function{1} = #1 \par
198 , subtitle-format  : function{1} = #1
199 , quote-format     : function{1} = #1

```

Not clear where this key should live and if it really makes sense to offer variations on the individual heading levels as in \UseStructureName{sec/???/number}.

```

200 , number-tag      : tokenlist = heading-number % \UseStructureName{sec/???}
201 , headformat-instance : tokenlist = std
202 , contents-extra : tokenlist =
203 }

```

`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`.

```

204 \DeclareTemplateInterface{heading}{runin}{9}
205 {
206 , name          : tokenlist
207 , parent-name    : tokenlist
208 , reset-counter  : tokenlist
209 , level          : integer = 0
210 , force-unnumbered : boolean = false
211 , placement      : choice {page , top , normal , ragged } = normal
212 , column-spanning : boolean = false
213 , mark-cmd       : function(1) =
214 , para-indent    : choice { true , false } = false
215 , before-vspace  : skip = Opt
216 , penalty        : integer = \c_max_int
217 , after-penalty-vspace : skip = Opt
218 , after-hspace   : skip = Opt
219 , start-code     : tokenlist = % no default values!
220 , final-code     : tokenlist = % no default values!
221 , prefix         : tokenlist = \NoValue
222 , punct         : tokenlist =
223 , heading-decls  : tokenlist = \normalfont

```

```

224 , prefix-decls      : tokenlist =
225 , number-decls      : tokenlist =
226 , title-decls       : tokenlist =
227 , subtitle-decls    : tokenlist =
228 , quote-decls       : tokenlist =
229 , heading-format     : function{1} = #1
230 , prefix-format      : function{1} = #1
231 , number-format      : function{1} = #1
232 , punct-format       : function{1} = #1

```

We have one difference in the defaults: in runin headings everything has to stay in horizontal mode so we do not append `\par` in the default value for `title-format`.

```

233 , title-format       : function{1} = #1
234 , subtitle-format    : function{1} = #1
235 , quote-format       : function{1} = #1
236 , number-tag         : tokenlist = heading-number % \UseStructureName{sec/???}
237 , headformat-instance : tokenlist = std
238 , contents-extra      : tokenlist =
239 }

```

11.7.3 headformat templates interfaces

We have three template: display, hang and runin.

headformat display (*templ.*) The `display` template produces a heading in which the heading elements form a single block, potentially with separators that separates some elements vertically from the other.

```

240 \DeclareTemplateInterface{headformat}{display}{6}
241 {
242 , heading-indent      : length = 0pt
243 %
244 , order               : commalist = {prefix,separator-a,?,number,punct,
245                                     separator-b,?,title,
246                                     separator-c, subtitle}
247 , separator-a         : tokenlist = \nobreakspace
248 , separator-b         : tokenlist = \par \nobreak \vspace{20pt}
249 , separator-c         : tokenlist = \
250 , separator-d         : tokenlist =
251 }

```

headformat hang (*templ.*) The `hang` template produces a heading where some elements are measured to determine a hanging indentation while the remaining ones are then typeset in that constrained space. This is why we have two order keys.

```

252 \DeclareTemplateInterface{headformat}{hang}{6}
253 {
254 , heading-indent      : length = 0pt
255 , order-hang          : commalist = {prefix,separator-a,?,number,punct,
256                                     separator-b,?}
257 , order               : commalist = {title, separator-c, subtitle}
258 , separator-a         : tokenlist = \nobreakspace
259 , separator-b         : tokenlist = \hspace{1em}
260 , separator-c         : tokenlist = \
261 , separator-d         : tokenlist =
262 }

```

`headformat runin (templ.)` The `runin` template produces a heading which is horizontally oriented and text following the heading will continue on the same line as the heading.

```

263 \DeclareTemplateInterface{headformat}{runin}{6}
264 {
265   , heading-indent      : length = Opt
266   , order               : commalist = {prefix,separator-a,?,number,punct,
267                                   separator-b,?,title,
268                                   separator-c,subtitle}
269   , separator-a         : tokenlist = \nobreakspace
270   , separator-b         : tokenlist = \hspace{1em}
271   , separator-c         : tokenlist = \
272   , separator-d         : tokenlist =
273 }

```

11.7.4 heading templates code

`heading display (templ.)`

```

274 \DeclareTemplateCode{heading}{display}{9}
275 {
276   , name                = \l__head_name_tl
277   , level               = \l__head_level_int
278   % next two not yet used
279   , parent-name         = \l__head_pname_tl
280   , reset-counter       = \l__head_reset_cnt_tl
281
282   , force-unnumbered    = \l__head_unnumbered_bool

```

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 usually implemented by setting `\l__head_start_code_tl` and `\l__head_final_code_tl` (exceptions are `normal` and `ragged`). These settings can be fine-tuned or overwritten with the keys `start-code` and `final-code`, if necessary.

```

282   , placement           = {
283     ,page               = \__head_debug_typeout:n{ A~ page~ heading }
284                       \tl_set:Nn \l__head_placement_tl { page }
285     ,top                = \__head_debug_typeout:n{ A~ top~ heading }
286                       \tl_set:Nn \l__head_placement_tl { top }
287     ,normal             = \__head_debug_typeout:n{ A~ normal~ heading }
288                       \tl_set:Nn \l__head_placement_tl { normal }
289     ,ragged             = \__head_debug_typeout:n{ A~ normal~ heading~ (ragged~ style) }
290                       \tl_set:Nn \l__head_placement_tl { ragged }
291                       }
292   , column-spanning     = \l__head_column_spanning_bool
293   , mark-cmd            = \__head_mark_cmd:n

```

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

```

294   , para-indent         = {
295     ,true               = \@afterindenttrue
296     ,false              = \@afterindentfalse
297   }
298   , before-vspace       = \l__head_before_skip

```

```

299 , penalty          = \l__head_penalty_int
300 , after-penalty-vspace = \l__head_after_penalty_skip
301 , after-vspace      = \l__head_after_skip
302 , start-code        = \l__head_start_code_tl
303 , final-code         = \l__head_final_code_tl

304 , prefix            = \l__head_prefix_tl
305 , punct              = \l__head_punct_tl

306 , headformat-instance = \l__head_headformat_instance_tl

307 , heading-decls      = \l__head_heading_decls_tl
308 , prefix-decls       = \l__head_prefix_decls_tl
309 , number-decls       = \l__head_number_decls_tl
310 , title-decls        = \l__head_title_decls_tl
311 , subtitle-decls     = \l__head_subtitle_decls_tl
312 , quote-decls        = \l__head_quote_decls_tl

313 , heading-format      = \__head_heading_format:n
314 , prefix-format       = \__head_prefix_format:n
315 , number-format       = \__head_number_format:n
316 , punct-format        = \__head_punct_format:n
317 , title-format        = \__head_title_format:n
318 , subtitle-format     = \__head_subtitle_format:n
319 , quote-format        = \__head_quote_format:n
320 %
321 , number-tag          = \l__head_number_tag_tl

322 , contents-extra      = \l__head_contents_extra_tl
323 }

324 {

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

```

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

```

326 \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}

327 \__head_show_arguments:nnnnnnnnn
328   {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

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

```

329 \SetKnownTemplateKeys{heading}{display}{#1}

```

If `column-spanning` is requested but we are not typesetting in two columns we turn it off to avoid using `\@topnewpage` as this would be wrong. Otherwise we check the placement value and adjust it if necessary.

```

330 \bool_if:NT \l__head_column_spanning_bool
331   { \if@twocolumn
332     \str_if_eq:VnF \l__head_placement_tl {top}
333     {
334       \__head_debug_typeout:n {column-spanning~ requires~
335         placement=top.~ Adjusted!}
336       \tl_set:Nn \l__head_placement_tl {top}
337     }
338   \else

```

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.

```

339         \bool_set_false:N \l__head_column_spanning_bool
340     \fi
341 }

```

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.

```

342 \tl_if_empty:oT \l__head_start_code_tl
343 { \str_case:VnF \l__head_placement_tl
344   {
345     { page }
346     { \tl_set:Nn \l__head_start_code_tl

```

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

Straight from the placement definition of `\part`.

```

347     { \if@openright
348       \cleardoublepage
349     \else
350       \clearpage
351     \fi
352     \thispagestyle{plain}%
353     \if@twocolumn
354       \onecolumn
355       \@tempswatrue
356     \else
357       \@tempswafalse
358     \fi
359     \null\vfil
360   }
361 }
362 { top }
363 { \tl_set:Nn \l__head_start_code_tl
364   { \if@openright\cleardoublepage\else\clearpage\fi
365     \thispagestyle{plain}%
366     \global\@topnum\z@
367   }
368 }

```

Ragged headings (i.e., those that leave a previous page ragged bottom if they generate a page break) are produced by replacing `\addpenalty` with a version that also adds `\vfil` and `\vfilneg`. This is handled by altering `\sec_add_penalty_with_possible_fil:n` which after use resets itself to `\addpenalty`.

```

369     { ragged }
370     { \cs_set_eq:NN \sec_add_penalty_with_possible_fil:n
371       \sec_add_penalty_with_fil:n
372     }
373   }
374   { \tl_clear:N \l__head_start_code_tl }
375 }
376 %
377 \tl_if_empty:oT \l__head_final_code_tl
378 { \str_case:VnF \l__head_placement_tl
379   {
380     { page }
381     { \tl_set:Nn \l__head_final_code_tl

```

```

382         { \vfil\newpage
383           \if@twoside
384             \if@openright
385               \null
386               \thispagestyle{empty}%
387             \newpage
388           \fi
389         \fi
390         \if@tempswa
391           \twocolumn
392         \fi
393       }
394     }
395   }
396   { \tl_set:Nn \l__head_final_code_tl { \@afterheading } }
397 }

```

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

```

398 \__head_determine_penalty:
399 \__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.

```

400 \__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.

```

401 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
402 \UseTaggingSocket{sec/begin}
403   {\int_use:N\l__head_level_int}
404   {tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}
405 }

```

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

```

406 \__head_debug_typeout:n{use~ 'headformat'~instance::~
407   \l__head_headformat_instance_tl }
408 \UseTaggingSocket{sec/title/begin}{\int_use:N\l__head_level_int}{#3}}

```

As long as we don't have a decent interface for the OR we have to use `\@topnewpage` for getting spanning headings (which means we are really using a top float box. That in turn means we have to get all the vertical spacing and so inside this box so there is a lot of back and forth based on the value of `\l__head_column_spanning_bool` for now.

```

409 \use:e {
410   \bool_if:NT \l__head_column_spanning_bool
411   { \exp_not:n {
412     \@topnewpage [
413       \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
414       { \vspace* \l__head_after_penalty_skip }
415     }
416   }
417   \UseInstance{headformat} { \l__head_headformat_instance_tl }

```

```

418         { \exp_not:o \UnusedTemplateKeys }
419         { \exp_not:o { \l__head_prefix_tl } }
420         { \exp_not:o { \l__head_number_tl } }
421         { \exp_not:n { #3 } }
422         { \exp_not:o { \use_i:nn #9 } }
423         { \exp_not:o { \use_ii:nn #9 } }

424     \bool_if:NT \l__head_column_spanning_bool
425     {
426         \skip_vertical:N \l__head_after_skip

```

Add the final vspace after the heading and close the `\@topnewpage` which has an optional argument.

```

427     ]
428   }
429 }

430 \UseTaggingSocket{sec/title/end}
431 %
432 % --- handle marks, toc-entry, bookmark, nameref, and label
433 %
434 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
435 %
436 % --- post-heading handling (vertical)

```

Restore `secnumdepth`

```

437 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
438 \par

```

If we have a spanning heading then the vertical skip is handled inside `\@topnewpage` and not here.

```

439 \bool_if:NF \l__head_column_spanning_bool
440 { \nobreak
441   \skip_vertical:N \l__head_after_skip
442 }
443 % --- prepare next paragraph (defaults to \cs{@afterheading}
444 %
445 \l__head_final_code_tl
446 %
447 \ignorespaces
448 }

```

`\sec_add_penalty_with_fil:n` The command `\sec_add_penalty_with_fil:n` adds a penalty surrounded by stretchable glue like the plain TeX `\filbreak` command. After use it sets `\sec_add_penalty_with_possible_fil:n` back to `\addpenalty`.

They have public names so that they can be used in other heading templates outside of this module.

```

449 \cs_new_protected:Npn \sec_add_penalty_with_fil:n #1 {
450   \__head_debug_typeout:n{apply~ ragged~ bottom~ fil}

```

Next code is straight from `\addpenalty` except that the generated `\penalty` is surrounded by `\vfil` and `\vfilneg`.

```

451   \addpenaltywithfil { #1 }

```

Then reset so that next time `\addpenalty` is used again.

```

452 \cs_set_eq:NN
453   \sec_add_penalty_with_possible_fil:n
454   \addpenalty
455 }

```

By default `\sec_add_penalty_with_possible_fil:n` is the same as `\addpenalty`. If you set it to `\sec_add_penalty_with_fil:n` then it uses the `filbreak` mechanism and afterwards resets itself to its default definition.

```

456 \cs_set_eq:NN
457   \sec_add_penalty_with_possible_fil:n
458   \addpenalty

```

(End of definition for `\sec_add_penalty_with_fil:n` and `\sec_add_penalty_with_possible_fil:n`. These functions are documented on page ??.)

```

\__kernel_add_penalty:n
  \addpenalty
  \addpenaltywithfil

```

Next code is for inclusion in the kernel. It is basically the code from L^AT_EX's `\addpenalty` but with a slightly different argument so that you have to supply `\penalty #1\relax` to mimic `\addpenalty` and `\vfil \penalty #1\vfilneg` to produce a penalty generating a filbreak, i.e., a break where the previous page is ragged bottom.

```

459 \cs_new_protected:Npn \__kernel_add_penalty:n #1 {

```

Don't add anything at the start of a minipage or when `@nobreak` is set (i.e., directly after a heading).

```

460   \mode_if_horizontal:T
461   { \mode_if_inner:TF { \@LRmoderr }{ \par } }
462   \legacy_if:nF { @minipage }
463   { \legacy_if:nF { @nobreak }

```

Do everything in a group so that the temp variables are restored afterwards in case they are used at the outside.

```

464     { \group_begin:

```

The rest is also from `\addpenalty` except that we have to give `\penalty` explicitly as part of the argument.

If there wasn't any previous skip or only a zero-sized one simply set the penalty.

```

465       \skip_set_eq:NN \l_tmpa_skip \tex_lastskip:D
466       \dim_compare:nNnTF \l_tmpa_skip = \c_zero_dim
467       { #1 }

```

Otherwise copy the last skip value also into `\l_tmpb_skip` using T_EX's fast direct assignment.

```

468       { \skip_set_eq:NN \l_tmpb_skip \l_tmpa_skip

```

Then add to it the current `\prevdepth` unless it is `-1000pt` (which means it should be suppressed) or if it is unusually large, in which case add `\maxdepth` instead. We remember that value because it will be reused below.

```

469       \dim_set:Nn \l_tmpa_dim
470       { \dim_compare:nNnTF \prevdepth > \maxdepth
471         \maxdepth
472         { \dim_compare:nNnTF \prevdepth = { -1000pt }
473           \c_zero_dim

```

```

474             \prevdepth
475         }
476     }
477     \skip_add:Nn \l_tmpb_skip \l_tmpa_dim

```

This is the amount we have to back up in order to position us vertically where the bottom of the page would be if a natural break would happen.

```

478     \vskip -\l_tmpb_skip

```

That is then the point where we have to add the explicit penalty or the filbreak code.

```

479         #1

```

After the penalty (and after a possible `\vfилneg`) we have to re-add what we back up but that isn't as trivial as one might think.

First we have to move forward by whatever amount we backed up due to `\prevdepth` because going forward the `\prevdepth` will be zero.

```

480     \dim_compare:nNnF \l_tmpa_dim = \c_zero_dim
481     { \vskip \l_tmpa_dim }

```

Finally we have to reinsert what we saved as `\lastskip`, because that is what any following `\addvspace` should see to make its calculations from.

```

482     \vskip \l_tmpa_skip
483 }
484 \group_end:
485 }
486 }
487 }

```

Reimplement `\addpenalty` using the above macro:

```

488 \cs_set_protected:Npn \addpenalty #1 {
489   \__kernel_add_penalty:n { \penalty #1 \scan_stop: }
490 }

```

And a new command to add a penalty with some `\vfil` around it. They cancel each other if no break is taken.

```

491 \cs_new_protected:Npn \addpenaltywithfil #1 {
492   \__kernel_add_penalty:n { \vfil \penalty #1 \vfилneg }
493 }

```

(End of definition for `\__kernel_add_penalty:n`, `\addpenalty`, and `\addpenaltywithfil`. These functions are documented on page ??.)

`\if@openright`

```

494 \AddToHook{begindocument}{
495   \ifcsname if@openright\endcsname
496   \else

```

Need to hide this a little if it is in fact already defined!

```

497     \expandafter\newif\csname if@openright\endcsname
498     \fi
499 }

```

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

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

```

500 \DeclareTemplateCode{heading}{runin}{9}
501 {
502   , name           = \l__head_name_tl
503   , level          = \l__head_level_int
504   % next two not yet used
505   , parent-name    = \l__head_pname_tl
506   , reset-counter  = \l__head_reset_cnt_tl

507   , force-unnumbered = \l__head_unnumbered_bool

```

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

```

508   , placement      = {
509     page           = \typeout{^^JA~ runin~ page~ placement~ heading~makes~no~sense
510                       (top-used)}
511     \tl_set:Nn \l__head_placement_tl { top }
512     ,top           = \__head_debug_typeout:n{ A~ top~ heading }
513     \tl_set:Nn \l__head_placement_tl { top }
514     ,normal        = \__head_debug_typeout:n{ A~ normal~ heading }
515     \tl_set:Nn \l__head_placement_tl { normal }
516     ,ragged        = \__head_debug_typeout:n{ A~ normal~ heading~ (ragged~ style) }
517     \tl_set:Nn \l__head_placement_tl { ragged }
518   }

519   , column-spanning = \l__head_column_spanning_bool

520   , 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.

```

521   , para-indent     = {
522     ,true           = \typeout{para-indent~ setting~ ignored}
523     ,false          = \typeout{para-indent~ setting~ ignored}
524   }
525   , before-vspace   = \l__head_before_skip
526   , penalty         = \l__head_penalty_int
527   , after-penalty-vspace = \l__head_after_penalty_skip
528   , after-hspace    = \l__head_after_skip
529   , start-code      = \l__head_start_code_tl
530   , final-code      = \l__head_final_code_tl

531   , prefix          = \l__head_prefix_tl
532   , punct           = \l__head_punct_tl

533   , headformat-instance = \l__head_headformat_instance_tl

534   , heading-decls   = \l__head_heading_decls_tl

535   , prefix-decls    = \l__head_prefix_decls_tl
536   , number-decls    = \l__head_number_decls_tl
537   , title-decls     = \l__head_title_decls_tl
538   , subtitle-decls  = \l__head_subtitle_decls_tl
539   , quote-decls     = \l__head_quote_decls_tl

```

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

```

540 , heading-format = \__head_heading_format:n
541 , prefix-format  = \__head_prefix_format:n
542 , number-format  = \__head_number_format:n
543 , punct-format   = \__head_punct_format:n
544 , title-format   = \__head_title_format:n
545 , subtitle-format = \__head_subtitle_format:n
546 , quote-format   = \__head_quote_format:n
547 %
548 , number-tag      = \l__head_number_tag_tl

549 , contents-extra = \l__head_contents_extra_tl
550 }
551 { \__head_show_arguments:nnnnnnnnn
552   {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

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

```

553 \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}

```

define `\theheading`

```

554 \tl_set_eq:Nc \theheading { the \l__head_name_tl }
555 \SetKnownTemplateKeys{heading}{runin}{#1}

```

For now we don't support column-spanning in this template.

```

556 \bool_if:NT \l__head_column_spanning_bool
557   { \__head_debug_typeout:n {column-spanning~ is~
558     not~ (yet)~ supported~ for~ runin~ headings!}
559     \bool_set_false:N \l__head_column_spanning_bool
560   }

561 \tl_if_empty:oT \l__head_start_code_tl
562   { \str_case:Vn \l__head_placement_tl
563     {
564       { top } { \tl_set:Nn \l__head_start_code_tl {\clearpage} }

565       { ragged }
566       { \cs_set_eq:NN \sec_add_penalty_with_possible_fil:n
567         \sec_add_penalty_with_fil:n
568       }
569     }

```

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

```

570 %   { \tl_clear:N \l__head_start_code_tl }
571 % }
572 %

```

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 ...

```

573 % \tl_if_empty:oT \l__head_final_code_tl
574 % { \str_case:VnF \l__head_placement_tl
575 %   {
576 %     { top } { \tl_clear:N \l__head_final_code_tl }
577 %   }
578 %   { \tl_clear:N \l__head_final_code_tl }
579 % }

```

```

580 \__head_determine_penalty:
581 \__head_determine_number_typesetting:N #2
582 \__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 para tagging here as run-in titles typeset the heading in \everypar.

```

583 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
584 \UseTaggingSocket{sec/begin}
585     {{\int_use:N\l__head_level_int}
586      {tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}
587     }
588 \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
589 \def \@svsechd {
590     \__head_debug_typeout:n{use~ 'headformat'~instance:~
591                          \l__head_headformat_instance_tl }
592     \use:e {
593         \UseInstance{headformat} { \l__head_headformat_instance_tl }
594         { \exp_not:o \UnusedTemplateKeys }
595         { \exp_not:o { \l__head_prefix_tl } }
596         { \exp_not:o { \l__head_number_tl } }
597         { \exp_not:n { #3 } }
598         { \exp_not:o { \use_i:nn #9 } }
599         { \exp_not:o { \use_ii:nn #9 } }
600     }
601     \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}

```

Restore secnumdepth

```

602     \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
603 }
604 \@nobreakfalse
605 \global\@noskipsectrue
606 \everypar{%
607     \if@noskipsec
608         \global\@noskipsecfalse
609         {\setbox\z@\lastbox}
610         \clubpenalty\@M
611         \@svsechd
612         \unskip

```

This tagging socket starts the “paragraph” after the run-in heading

```

613     \UseTaggingSocket{sec/title/split}
614     \skip_horizontal:N \l__head_after_skip
615     \else
616         \clubpenalty \@clubpenalty
617         \everypar{}%
618     \fi
619 }

```

— prepare next paragraph (does nothing by default)

```

620 \l__head_final_code_tl
621 \ignorespaces
622 }

```

11.7.5 headformat templates code

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

```

623 \DeclareTemplateCode{headformat}{display}{6}
624 % args: keys,prefix,number,title,subtitle,quotation
625 {
626   , heading-indent    = \l__head_heading_indent_dim
627   , order             = \l__head_order_clist
628   , separator-a       = name {l__head_separator-a_tl}
629   , separator-b       = name {l__head_separator-b_tl}
630   , separator-c       = name {l__head_separator-c_tl}
631   , separator-d       = name {l__head_separator-d_tl}
632 }
633 {
634   \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

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

```

635   \SetTemplateKeys{headformat}{display}{#1}
636   \group_begin:

```

`\parboxes` can be used within the title of a heading (or withing it layout). `minipage` environments currently do not work if used directly in the title because `purify` can't handle that environment. But it can be used as part of the layout definition which is why we also add a suitable setting for it.

```

637   \AssignTaggingSocketPlug{parbox/before}{heading}
638   \AssignTaggingSocketPlug{minipage/before}{heading}
639   \AssignTaggingSocketPlug{para/restore}{heading}
640   \tagpdfparaOff
641 %
642   \tl_set:Nn \l__head_prefix_tl {#2}
643   \tl_set:Nn \l__head_number_tl {#3}
644   \tl_set:Nn \l__head_title_tl {#4}
645 %

```

TODO: revisit if `\normalcolor` is needed, if yes, use as `\SaveLastSkip \normalcolor \RestoreLastSkip`.

```

646   \normalfont
647   \interlinepenalty \@M
648   \l__head_heading_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!

```

649   \bool_if:NTF \l__head_unnumbered_bool
650   { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
651     { \skip_horizontal:N \l__head_heading_indent_dim
652       \MakeLinkTarget[\l__head_name_tl]{}
653     }
654     { \MakeLinkTarget[\l__head_name_tl]{}
655       \skip_horizontal:N \l__head_heading_indent_dim
656     }

```

```

657     }
658     { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
659       { \skip_horizontal:N \l__head_heading_indent_dim
660         \MakeLinkTarget{\l__head_name_tl}
661       }
662       { \MakeLinkTarget{\l__head_name_tl}
663         \skip_horizontal:N \l__head_heading_indent_dim
664       }
665     }
666   %
667   \__head_heading_format:n {
668     \template_process_order_clist:nnn
669     { head }{ order }
670     { number, prefix, punct, separator-a,
671       separator-b, separator-c, separator-d, title, subtitle }
672   }
673   \par
674   \group_end:
675 }

```

The the plugs we assigned to the tagging sockets in the template need to be defined.

```

676 \NewTaggingSocketPlug{minipage/before}{heading}
677 { \tag_mc_end_push:
678   \bool_if:NT \l__tag_para_bool {\tag_struct_end:}
679   \tag_struct_begin:n{tag=\UseStructureName{sec/minipage}}
680 }

681 \NewTaggingSocketPlug{parbox/before}{heading}
682 { \tag_mc_end_push:
683   \bool_if:NT \l__tag_para_bool {\tag_struct_end:}
684   \tag_struct_begin:n{tag=\UseStructureName{sec/parbox}}
685 }

686 \NewTaggingSocketPlug{para/restore}{heading}

```

We have only structure names on heading levels for this, so for now I use `heading-fragment` directly. Should probably become `sec/fragment`.

```

687 { \AssignStructureRole{para/textblock}{heading-fragment}
688   \bool_set_true:N\l__tag_para_flattened_bool
689   \tagpdfparaOn
690 }

```

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

```

691 \DeclareTemplateCode{headformat}{hang}{6}
692 % args: keys,prefix,number,title,subtitle,quotation
693 {
694   , heading-indent      = \l__head_heading_indent_dim
695   , order-hang          = name {l__head_order-hang_clist}
696   , order               = \l__head_order_clist
697   , separator-a         = name {l__head_separator-a_tl}
698   , separator-b         = name {l__head_separator-b_tl}
699   , separator-c         = name {l__head_separator-c_tl}
700   , separator-d         = name {l__head_separator-d_tl}
701 }

```

```

702 {
703   \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

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

```

704   \SetTemplateKeys{headformat}{hang}{#1}

705   \group_begin:
706   %
707   \AssignTaggingSocketPlug{parbox/before}{noop}
708   \AssignTaggingSocketPlug{parbox/after}{noop}
709   \AssignTaggingSocketPlug{para/restore}{noop}
710   \tagpdfparaOff
711   %
712   \tl_set:Nn \l__head_prefix_tl {#2}
713   \tl_set:Nn \l__head_number_tl {#3}
714   \tl_set:Nn \l__head_title_tl {#4}
715   %

716   \normalfont
717   \interlinepenalty \OM
718   \l__head_heading_decls_tl
719   \noindent
720   \setbox\@tempboxa\hbox{{

```

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.

```

721     \bool_if:NTF \l__head_unnumbered_bool
722     { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
723       { \skip_horizontal:N \l__head_heading_indent_dim
724         \MakeLinkTarget[\l__head_name_tl]{}
725       }
726       { \MakeLinkTarget[\l__head_name_tl]{}
727         \skip_horizontal:N \l__head_heading_indent_dim
728       }
729     }
730     { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
731       { \skip_horizontal:N \l__head_heading_indent_dim
732         \MakeLinkTarget{\l__head_name_tl}
733       }
734       { \MakeLinkTarget{\l__head_name_tl}
735         \skip_horizontal:N \l__head_heading_indent_dim
736       }
737     }
738     \template_process_order_clist:nnn
739     { head }{ order-hang }
740     { number, prefix, punct, separator-a,
741       separator-b, separator-c, separator-d, title, subtitle }
742   }}
743   \hangindent \wd\@tempboxa
744   \box\@tempboxa
745   \template_process_order_clist:nnn
746   { head }{ order }
747   { number, prefix, punct, separator-a,
748     separator-b, separator-c, separator-d, title, subtitle }

```

```

749     \par
750   \group_end:
751 }
752

```

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

```

753 \DeclareTemplateCode{headformat}{runin}{6}
754   % args: keys,prefix,number,title,subtitle,quotation
755   {
756     , heading-indent      = \l__head_heading_indent_dim
757     , order               = \l__head_order_clist
758     , separator-a         = name {l__head_separator-a_tl}
759     , separator-b         = name {l__head_separator-b_tl}
760     , separator-c         = name {l__head_separator-c_tl}
761     , separator-d         = name {l__head_separator-d_tl}
762   }
763   {
764     \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

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

```

765   \SetTemplateKeys{headformat}{hang}{#1}

766   \group_begin:
767   %
768     \AssignTaggingSocketPlug{parbox/before}{noop}
769     \AssignTaggingSocketPlug{parbox/after}{noop}
770     \AssignTaggingSocketPlug{para/restore}{noop}
771     \tagpdfparaOff
772   %
773     \tl_set:Nn \l__head_prefix_tl {#2}
774     \tl_set:Nn \l__head_number_tl {#3}
775     \tl_set:Nn \l__head_title_tl {#4}

776   \normalfont

777   \interlinepenalty \OM
778   \l__head_heading_decls_tl

```

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

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

```

779   \bool_if:NTF \l__head_unnumbered_bool
780   { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
781     { \skip_horizontal:N \l__head_heading_indent_dim
782       \MakeLinkTarget[\l__head_name_tl]{}
783     }
784     { \MakeLinkTarget[\l__head_name_tl]{}
785       \skip_horizontal:N \l__head_heading_indent_dim
786     }
787   }
788   { \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
789     { \skip_horizontal:N \l__head_heading_indent_dim

```

```

790         \MakeLinkTarget{\l__head_name_tl}
791     }
792     { \MakeLinkTarget{\l__head_name_tl}
793       \skip_horizontal:N \l__head_heading_indent_dim
794     }
795 }
796 \template_process_order_clist:nnn
797   { head }{ order }
798   { number, prefix, punct, separator-a,
799     separator-b, separator-c, separator-d, title, subtitle }
800 \group_end:
801 }

```

11.7.6 Internal commands used by the template code

`\__head_determine_penalty:`

```

802 \cs_new:Npn \__head_determine_penalty: {

```

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

```

803   \int_compare:nNt \l__head_penalty_int = \c_max_int
804   { \int_set:Nn \l__head_penalty_int \@secpenalty }
805 }

```

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

`\__head_determine_number_typesetting:N`

```

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

```

If `force-unnumbered` was set to true (i.e., `\l__head_unnumbered_bool`) we don't do numbering. If that is not the case then 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.

```

807   \bool_if:NF \l__head_unnumbered_bool
808   { \bool_set:Nn \l__head_unnumbered_bool
809     { \bool_lazy_or_p:nn
810       { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
811       { \bool_if_p:N #1 }
812     }
813   }

```

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)

```

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

```

and we set `\l__head_number_tl`, `\l__head_prefix_tl`, and `\l__head_punct_tl` to do nothing.

```

816   \tl_set:Nn \l__head_number_tl { \NoValue }
817   \tl_set:Nn \l__head_prefix_tl { \NoValue }
818   \tl_set:Nn \l__head_punct_tl { \NoValue }
819 }

```

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

```

820     {
821       \@kernel@refstepcounter{ \l__head_name_tl }
822       \tl_set:Nn \l__head_number_tl { \theheading }
823     }
824   }

```

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

`\__head_vertical_before_spacing:`

```

825 \cs_new:Npn \__head_vertical_before_spacing: {
826   \tl_if_blank:VTF \l__head_start_code_tl
827   { \if@noskipsec \leavevmode \fi
828     \par
829     \if@nobreak
830       \everypar{}
831     \else

```

Normally the penalty is added with `\addpenalty` but in some cases (`placement=ragged`) we want to use a variant of `\filbreak` instead. So we use `\sec_add_penalty_with_possible_fil:n` which is either equal to `\addpenalty` or to `\sec_add_penalty_with_fil:n` depending on the setup.

```

832       \sec_add_penalty_with_possible_fil:n
833       \l__head_penalty_int
834       \addvspace \l__head_before_skip

```

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

```

835       \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
836       { \vspace* \l__head_after_penalty_skip }
837     \fi
838   }
839   {
840     \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. And we do nothing at all when we have a spanning heading. Then this `vspace` is done inside `\@topnewpage`.

```

841     \bool_if:NF \l__head_column_spanning_bool
842     {
843       \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
844       { \vspace* \l__head_after_penalty_skip }
845     }
846   }
847 }

```

(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.

```

848 \newcommand\addcontentslinebookmark[3]{}
849 \newcommand\addcontentslinebookmarkOff{}
850 \newcommand\addcontentslinebookmarkReset{}
851 \providecommand\texorpdfstring[2]{#1}

```

This can go once hyperref is updated (ufi,2026-04-21):

```

852 \AddToHook{package/hyperref/after}
853 {
854   \RenewCommandCopy\addcontentslinebookmark
855     \Hy@addcontentsline@addbookmark
856   \renewcommand\addcontentslinebookmarkOff
857     { \ifHy@bookmarks
858       \let\addcontentslinebookmarkReset
859         \Hy@bookmarkstrue
860       \else
861         \let\addcontentslinebookmarkReset\relax
862       \fi
863       \Hy@bookmarksfalse
864     }
865 }

```

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

_head_handle_marks_etc:nmmnn

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

```

866 \cs_new:Npn \_head\_handle\_marks\_etc:nmmnn #1#2#3#4#5 {
867   %
868   \tl_set:Nn\@currentlabelname{#4}
869   \IfBlankF {#2}
870     { \_head\_mark\_cmd:n { #2 } }
871   \IfBlankTF {#1}

```

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

```

872   { \IfBlankF{#3}
873     {
874       \addcontentslinebookmark{toc}{\l__head\_name\_tl}
875       {
876         \bool_if:NF \l__head\_unnumbered\_bool
877         {
878           \protect\numberline{ \use:c{ the \l__head\_name\_tl } }
879         }
880         #3
881       }
882     }
883   }

```

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.

```

884   { \IfBlankT{#3}{\addcontentslinebookmarkOff}
885     \addcontentsline{toc}{ \l__head\_name\_tl }

```

```

886     {
887         \bool_if:NF \l__head_unnumbered_bool
888         {
889             \protect\numberline{ \use:c{ the \l__head_name_tl } }
890         }

```

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

```

891         \texorpdfstring{#1}{#3}
892     }
893     \addcontentslinebookmarkReset
894 }

```

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

```

895     \l__head_contents_extra_tl

```

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

```

896     \__head_debug_typeout:n{--->~label(s):~ \exp_not:n{#5}}
897     #5
898 }

```

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

11.8 Support for legacy classes and packages

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

```

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

```

then this gets overwritten by every class right now.

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

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

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

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

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

The situation with headings defined via `\secdef` is worse: we don’t have a nice handle as with `\@startsection` so there is no real way other than through static analysis

perhaps this should have a hook for packages

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.

11.8.1 Instances (sample/default definitions)

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

```

899 \DeclareInstance{heading}{part}{display}
900 {
901   , name           = part
902   , level          = -1
903   , placement      = page
904   , after-penalty-vspace = 0pt
905   , prefix         = \partname
906   , number-format  = \thepart
907   , heading-decls  = \centering\bfseries\huge
908   , title-decls    = \Huge
909   , headformat-instance = part
910   , mark-cmd       = \partmark {#1}
911 }
```

`heading chapter` (*inst.*)

```

912 \DeclareInstance{heading}{chapter}{display}
913 {
914   , name           = chapter
915   , level          = 0
916   , placement      = top
917   , column-spanning = true
918   , after-penalty-vspace = 50pt
919   , after-vspace    = 40pt
920   , prefix         = \@chapapp
921   , number-format  = \thechapter
922   , heading-decls  = \raggedright \parindent0pt \bfseries \huge
923   , title-decls    = \Huge
924   , headformat-instance = chapter
925   , mark-cmd       = \chaptermark {#1}
926   , contents-extra = \addtocontents{lof}{\addvspace{10pt}}
927                   \addtocontents{lot}{\addvspace{10pt}}
928 }
```

`\@chapapp` To improve compatibility with classes that use `\secdef` but don't provide a definition for `\@chapapp` used in the `prefix` key above, we define this command if necessary. Otherwise such a class would error if `\chapter` is used.

```

929 \AtBeginDocument{
930   \cs_if_exist:NF \@chapapp { \cs_new:Npn \@chapapp {Chapter} } }

```

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

`heading section (inst.)`

```

931 \DeclareInstance{heading}{section}{display}
932 {
933   , name           = section
934   , level          = 1
935   , mark-cmd       = \sectionmark {#1}
936   , before-vspace  = 3.5ex plus 1ex minus .2ex
937   , after-vspace   = 2.3ex plus .2ex
938   , heading-decls  = \normalfont\Large\bfseries
939   , headformat-instance = section
940 }

```

`heading subsection (inst.)`

```

941 \DeclareInstance{heading}{subsection}{display}
942 {
943   , name           = subsection
944   , parent-name    = section
945   , level          = 2
946   , mark-cmd       = \subsectionmark {#1}
947   , before-vspace  = 3.25ex plus 1ex minus .2ex
948   , after-vspace   = 1.5ex plus .2ex
949   , heading-decls  = \normalfont\large\bfseries
950   , headformat-instance = subsection
951 }

```

`heading subsubsection (inst.)`

```

952 \DeclareInstance{heading}{subsubsection}{display}
953 {
954   , name           = subsubsection
955   , parent-name    = subsection
956   , level          = 3
957   , before-vspace  = 3.25ex plus 1ex minus .2ex
958   , after-vspace   = 1.5ex plus .2ex
959   , heading-decls  = \normalfont\normalsize\bfseries
960   , headformat-instance = subsubsection
961 }

```

`heading paragraph (inst.)`

```

962 \DeclareInstance{heading}{paragraph}{runin}
963 {
964   , name           = paragraph
965   , parent-name    = subsubsection
966   , level          = 4
967   , before-vspace  = 3.25ex \@plus1ex \@minus.2ex
968   , after-hspace   = 1em

```

```

969 , heading-decls = \normalfont\normalsize\bfseries
970 , mark-cmd      = \paragraphmark {#1}
971 , headformat-instance = paragraph
972
973 }

```

heading subparagraph (*inst.*)

```

974 \DeclareInstance{heading}{subparagraph}{runin}
975 {
976   , name          = subparagraph
977   , parent-name   = paragraph
978   , level         = 5
979   , before-vspace = 3.25ex \@plus1ex \@minus .2ex
980   , after-hspace  = 1em
981   , heading-decls = \normalfont\normalsize\bfseries
982   , mark-cmd      = \subparagraphmark {#1}
983   , headformat-instance = subparagraph
984
985 }

```

headformat part (*inst.*) The headformat instances for part and chapter use the display template with identical
headformat chapter (*inst.*) settings by default, so one is made a copy of the other to speed up loading.

```

986 \DeclareInstance{headformat}{part}{display}
987 {
988   , heading-indent = 0pt
989   , separator-a    = \nobreakspace           % between prefix and number
990   , separator-b    = \par \nobreak \vspace{20pt} % between label block and title
991 }
992 \DeclareInstanceCopy{headformat}{chapter}{part}

```

headformat std (*inst.*) Similarly, by default the instances for section, subsection, and subsubsection are all using
headformat section (*inst.*) the hang template and the same key values as the std instance, so they are all made a
headformat subsection (*inst.*) copy of that one.
headformat subsubsection (*inst.*)

```

993 \DeclareInstance{headformat}{std}{hang}
994 {
995   , heading-indent = 0pt
996 }
997 \DeclareInstanceCopy{headformat}{section}{std}
998 \DeclareInstanceCopy{headformat}{subsection}{std}
999 \DeclareInstanceCopy{headformat}{subsubsection}{std}

```

headformat paragraph (*inst.*) In contrast, paragraph and subparagraph differ even though both use the runin template.
headformat subparagraph (*inst.*)

```

1000 \DeclareInstance{headformat}{paragraph}{runin}
1001 {
1002   , heading-indent = 0pt
1003 }
1004 \DeclareInstance{headformat}{subparagraph}{runin}
1005 {
1006   , heading-indent = \parindent
1007 }

```

adformat section-special (*inst.*)

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

```

1008 \DeclareInstance{headformat}{section-special}{hang}
1009 {
1010   , heading-indent = 4em
1011   %%% , title-format = {#1!} % no longer
1012 }

```

11.8.2 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 `heading-decls` and to `title-format`.

```

1013 \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.

```

1014   \IfInstanceExistsF{heading}{#1-\@startsection}{
1015     \__head_debug_typeout:n{Info:~ setting~ up~ instances~ for~
1016       legacy~ \string\@startsection}

```

`\@startsection` supported the use of a macro with one argument as the last token in its `<style>` argument, for example, `\MakeUppercase` to make the whole title uppercase. To support that we need to take a look at `#6` and if that ends in such a macro split it off, so that we can put the first tokens into `heading-decls` and this last token into `title-format`. This is what the next call prepares for:

```

1017   \__head_prep_decls_and_format_from_tl:nNN {#6}
1018   \l__head_heading_decls_tl
1019   \l__head_title_format_tl

```

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`.

```

1020   \@tempskipa #4\relax
1021   \@afterindenttrue
1022   \ifdim \@tempskipa <\z@
1023     \@tempskipa -\@tempskipa
1024   \@afterindentfalse
1025   \fi

```

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

```

1026 \@tempskipb #5\relax
1027 \ifdim \@tempskipb>\z@
1028 \use:e {
1029 \DeclareInstance{heading}{#1-@startsection}{display}{
1030 , name = #1
1031 , level = #2
1032 , mark-cmd = \exp_not:c {#1mark} {##1}
1033 , before-vspace = \the\@tempskipa
1034 , after-vspace = \the\@tempskipb
1035 , para-indent = \if@afterindent true \else false\fi

```

Argument #6 contains the declarations for the whole heading but it is allowed that the last token is a macro with one argument that receives the heading text as its argument. This is why we have split off the last token above, in case it contained a macro with one argument. That token is then used as the `title-format`. We end this key value in `\par` if we are in a display instance.

```

1036 , heading-decls = \exp_not:o \l__head_heading_decls_tl
1037 , title-format = \exp_not:o \l__head_title_format_tl {##1}
1038 \exp_not:N \par
1039 , headformat-instance = #1-@startsection
1040 }}
1041 \DeclareInstance{headformat}{#1-@startsection}{hang}{heading-indent = #3}

```

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

```

1042 \else
1043 \@tempskipb=-\@tempskipb
1044 \use:e {
1045 \DeclareInstance{heading}{#1-@startsection}{runin}{
1046 , name = #1
1047 , level = #2
1048 , mark-cmd = \exp_not:c {#1mark} {##1}
1049 , before-vspace = \the\@tempskipa
1050 , after-hspace = \the\@tempskipb
1051 , heading-decls = \exp_not:o \l__head_heading_decls_tl

```

In a runin instance `title-format` should not end in `\par`!

```

1052 , title-format = \exp_not:o \l__head_title_format_tl {##1}
1053 , headformat-instance = #1-@startsection
1054 }}
1055 \DeclareInstance{headformat}{#1-@startsection}{runin}{heading-indent = #3}
1056 \fi
1057 }
1058 \ParseLaTeXeHeading {#1-@startsection}{#7}{#8}{#9}
1059 }

```

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

Examine #1. If the last token is a macro with one argument put it in #3 and the remainder of #1 into #2. Otherwise, put `\use:n` in #3 and all of #1 in #2.

```

1060 \cs_new:Npn \__head_prep_decls_and_format_from_tl:nNN #1#2#3
1061 { \tl_set:Nn #3 { \use:n }

```

```

1062     \exp_args:Ne
1063     \__head_prep_decls_and_auxi:nnNN
1064         { \tl_reverse:n {#1} } {#1} #2 #3
1065 }

1066 \cs_new:Npn \__head_prep_decls_and_auxi:nnNN #1#2#3#4
1067 { \tl_if_head_is_N_type:nTF {#1}
1068   { \__head_prep_decls_and_auxii:nNwNN {#2} #1 \q_stop #3 #4 }
1069   { \tl_set:Nn #3 {#2} }
1070 }

1071 \cs_new:Npn \__head_prep_decls_and_auxii:nNwNN #1#2#3 \q_stop #4 #5
1072 { \token_if_macro:NTF #2
1073   { \exp_args:Ne
1074     \__head_prep_decls_and_auxiii:nnNnNN
1075     { \cmd_arg_spec:N #2 } {#1} #2 {#3} #4 #5
1076   }
1077   { \tl_set:Nn #4 {#1} }
1078 }

1079 \cs_new:Npn \__head_prep_decls_and_auxiii:nnNnNN #1#2#3#4#5#6 {
1080   \bool_lazy_any:nTF
1081   {

```

This tests for macros with one argument and defined by `\def` or similar.

```

1082     { \str_if_eq_p:ee { \cs_parameter_spec:N #3 } { \c_hash_str 1 } }

```

For those defined with `\NewDocumentCommand` we need to look at the signature instead. We also accept macros with one optional argument first followed by a mandatory one.

```

1083     { \str_if_eq_p:nn {#1} { m } }
1084     { \str_if_eq_p:nn {#1} { +m } }
1085     { \str_if_eq_p:nn {#1} { 0{ } m } }
1086     { \str_if_eq_p:nn {#1} { 0{ } +m } }

```

Another possibility are macros declared with `\DeclareRobustCommand`.

```

1087     { \bool_lazy_and_p:nn
1088       { \cs_if_exist_p:c { \cs_to_str:N #3 \c_space_tl } }
1089       { \str_if_eq_p:ee
1090         { \cs_parameter_spec:c { \cs_to_str:N #3 \c_space_tl } }
1091         { \c_hash_str 1 }
1092       }
1093     }
1094     { \bool_lazy_and_p:nn
1095       { \cs_if_exist_p:c { \token_to_str:N #3 \c_space_tl } }
1096       { \str_if_eq_p:ee
1097         { \cs_parameter_spec:c { \token_to_str:N #3 \c_space_tl } }
1098         { [ \c_hash_str 1 ] \c_hash_str 2 }
1099       }
1100     }
1101 }
1102 { \tl_set:Ne #5 { \tl_reverse:n {#4} }
1103   \tl_set:Nn #6 {#3}
1104 }
1105 { \tl_set:Nn #5 {#2} }
1106 }

```

```

1107 \cs_generate_variant:Nn \cs_parameter_spec:N { c }
(End of definition for \__head_prep_decls_and_format_from_tl:nNN.)

```

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

```

1108 \NewCommandCopy\kernel@startsection\@startsection
1109 \AddToHook{class/after}[head/@startsection]
1110 {\RenewCommandCopy\@startsection\kernel@startsection}

```

11.8.3 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
1111 \RenewDocumentCommand\secdef{mmsO{#5}m}
1112 {
1113   \str_case:enF {\cs_to_str:N#1}
1114   {
1115     {@part}
1116     { \IfNoValueTF{#4}
1117       {\ParseLaTeXeHeading{part}{#3} {start-code=\relax} {#5}}
1118       {
1119         \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
1120         \ExpandArgs{nne}\ParseLaTeXeHeading{part}{#3}
1121         {start-code=\relax,\exp_not:o{\l__head_tmpa_tl}} {#5}
1122       }
1123       \@latex@warning{\noexpand\secdef~ detected~ in~ \noexpand\part
1124         command.\MessageBreak
1125         \noexpand\part~will~be~redefined~to~use~templates.\MessageBreak
1126         This~may~change~the~layout!}
1127       \DeclareDocumentCommand \part {s ={shorttitle}o m}
1128       { \ParseLaTeXeHeading {part} {##1} {##2} {##3} }
1129     }
1130     {@chapter}
1131     { \IfNoValueTF{#4}
1132       { \ParseLaTeXeHeading{chapter}{#3} {#4} {#5} }
1133       {
1134         \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
1135         \ExpandArgs{nno}\ParseLaTeXeHeading{chapter}{#3}
1136         {\l__head_tmpa_tl} {#5}
1137       }
1138       \@latex@warning{\noexpand\secdef~ detected~ in~
1139         \noexpand\chapter command.\MessageBreak
1140         \noexpand\chapter~ will~ be~ redefined~ to~ use~
1141         templates.\MessageBreak
1142         This~may~change~the~layout!}
1143       \DeclareDocumentCommand \chapter {s ={shorttitle}o m}
1144       { \ParseLaTeXeHeading {chapter} {##1} {##2} {##3} }
1145     }
1146   }
1147   {

```

```

1148 \tag_if_active:TF\@latex@error\@latex@warning
1149 {\noexpand\secdef~ with~ unknown~ argument~ \noexpand#1
1150 found.\MessageBreak
1151 The~command~can~not~be~adapted~on~the~fly~to~support~tagging.\MessageBreak
1152 The~heading~command~using~this~\noexpand\secdef~should~be~
1153 reimplemented\MessageBreak with~templates}{\}
1154 \IfBooleanTF{#3}{#2{#5}}{#1[#4]{#5}}
1155 }
1156 }

```

A helper command to reprocess the argument as key-val

```

1157 \NewDocumentCommand\__head_process_shorttitle:nN{={shorttitle}mm}
1158 { \tl_set:Nn#2{#1} }

```

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

11.8.4 Core L^AT_EX

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

```

1159 \AddToHook{class/article/after}[head/example]
1160 {
1161   \DeclareInstance{heading}{part}{display}
1162   {
1163     , name           = part
1164     , level          = -1
1165     , before-vspace  = 4ex
1166     , after-vspace   = 3ex
1167     , mark-cmd       = \partmark {#1}
1168     , prefix         = \partname
1169     , punct          =
1170     , heading-decls  = \raggedright\bfseries\Large
1171     , title-decls    = \huge
1172     , headformat-instance = part
1173   }
1174   \DeclareInstance{headformat}{part}{display}
1175   {
1176     , heading-indent = 0pt
1177     , separator-a     = \nobreakspace
1178     , separator-b     = \par \nobreak \vspace{20pt}
1179   }
1180   \DeclareDocumentCommand \part {s = {shorttitle}o m}
1181   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1182   \__head_setup_default_heading:
1183 }

1184 \AddToHook{class/report/after}[head/example]
1185 {
1186   \DeclareDocumentCommand \part {s = {shorttitle}o m}
1187   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1188   \DeclareDocumentCommand \chapter {s = {shorttitle}o m}
1189   {
1190     \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
1191   }

```

```

1192     \__head_setup_default_heading:
1193   }
1194   \AddToHook{class/book/after}[head/example]
1195   {
1196     \DeclareDocumentCommand \part {s ={\shorttitle}o m}
1197     { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1198     \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
1199     {
1200       \if@mainmatter
1201         \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
1202       \else
1203         \IfBooleanTF{#1}
1204         {% starred:
1205           \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
1206         }
1207         {\IfNoValueTF{#2}
1208          {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {\shorttitle={#3}} {#3}}
1209          {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}
1210        }
1211       \fi
1212     }
1213     \__head_setup_default_heading:
1214   }

1215   \cs_new_protected:Npn \__head_setup_default_heading:
1216   {

\section
1217     \DeclareDocumentCommand \section {s ={\shorttitle}o m}
1218     { \ParseLaTeXeHeading {section} {##1} {##2} {##3} }
(End of definition for \section. This function is documented on page ??.)

\subsection
1219     \DeclareDocumentCommand \subsection {s ={\shorttitle}o m}
1220     { \ParseLaTeXeHeading {subsection} {##1} {##2} {##3} }
(End of definition for \subsection. This function is documented on page ??.)

\subsubsection
1221     \DeclareDocumentCommand \subsubsection {s ={\shorttitle}o m}
1222     { \ParseLaTeXeHeading {subsubsection} {##1} {##2} {##3} }
(End of definition for \subsubsection. This function is documented on page ??.)

\paragraph
1223     \DeclareDocumentCommand \paragraph {s ={\shorttitle}o m}
1224     { \ParseLaTeXeHeading {paragraph} {##1} {##2} {##3} }
(End of definition for \paragraph. This function is documented on page ??.)

\subparagraph
1225     \DeclareDocumentCommand \subparagraph {s ={\shorttitle}o m}
1226     { \ParseLaTeXeHeading {subparagraph} {##1} {##2} {##3} }
1227   } %end of command
(End of definition for \subparagraph. This function is documented on page ??.)

1228 </package>

```

12 Issues and problems noticed along the way

12.1 Local `\DeclareInstance`

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

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

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

12.2 `\SetCurrentTemplateKeys`

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

12.3 O-expansion of `\UseInstance` arguments

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

```
\use:e {  
  \UseInstance{headformat} { \l_@@_headformat_instance_tl }  
    { \exp_not:o \UnusedTemplateKeys }  
    { \exp_not:o { \l_@@_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.

A Templates from the 2026-06-01 release of `LATEX`

In this appendix we provide the template implementations that I came up with in my first attempt (only renamed to `<name>-v1`). Eventually, we want to drop the `-v1` versions but for the next couple of releases we provide them alongside the new more flexible templates, so that there is no need to transitioning to the new templates in a rush.

A.1 Template documentation for -v1 templates

A.1.1 Templates of type heading

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., `heading-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.⁴

A.1.2 The heading template ‘`<all>`’

Attributes:

name (*tokenlist*) Referenceable 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.

prefix (*tokenlist*) A fixed string, such as “Chapter”, that can be used together with the number (if any) to form a “heading label”. Usage and placement is up to the template, so it could be placed after the number by the template (in which case it isn’t really a prefix). Default: `\NoValue`

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

⁴Maybe questionable

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

At the moment I have retained the L^AT_EX 2_ε 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>`.

A.1.3 The heading template ‘display-v1’

Attributes:

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

before-vspace (*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 (T_EX's largest integer) indicates that no penalty was set in which case `\@secpenalty` is used.
Default: `1073741823`

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

heading-decls (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Note that color commands (unless `luacolor` is used) introduce a break point before the heading and therefore one should either surround color commands with `\SaveLastSkip` and `\RestoreLastSkip` or to use the keys `prefix-decls`, `number-decls`, `title-decls`, etc. instead. Default: `\normalfont`

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

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

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

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

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

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

A.1.4 The heading template ‘runin-v1’

Attributes:

before-vspace (*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-vspace (*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-vspace (*skip*) Vertical space after the heading – ignored in `runin` headings so should be dropped! Default: `0pt`

heading-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`

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

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

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

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

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

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

A.1.5 Templates of type `headformat`

At the moment all templates of this type assume that placement of prefix, number, title is done in exactly this order. Anything else would require defining further templates.

TODO: consider a more versatile mechanism (like in theorem-like templates) to allow for more flexible placements without the need to define additional templates.

A.1.6 The `headformat` template ‘hang-v1’

Attributes:

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

title-format (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. The code takes an argument, e.g., `\MakeUppercase` Default: `<#1>`

prefix-number-sep (*tokenlist*) Token list that can be used in the assignment to a $\text{\TeX}$ dimension (can contain `em` or `ex` values) specifying the separation between title prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

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

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

A.1.7 The headformat template ‘runin-v1’

Attributes:

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

title-format (*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`

prefix-number-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 prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

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

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

Semantics & Comments: Implements a heading layout where number and title start on the same line but then continue with the following paragraph on the same line (or the next if the title overflows, i.e., the title is run-in. It is what $\LaTeX$ always used by default for `\paragraph` and `\subparagraph`.

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

A.1.8 The headformat template ‘display-v1’

Attributes:

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

title-format (*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: `\#1`

prefix-number-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 prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

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 L^AT_EX always used by default for `\chapter` and `\part`.

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

A.2 Implementation of -v1 templates

heading display-v1 (*templ.*)

```

1229 \DeclareTemplateInterface{heading}{display-v1}{9}
1230 {
1231   , name           : tokenlist
1232   , parent-name    : tokenlist
1233   , reset-counter  : tokenlist
1234   , level          : integer = 0
1235   , placement      : choice {page , top , normal } = normal
1236   , mark-cmd       : function(1) =
1237   , para-indent    : choice { true , false } = false
1238   , before-vspace  : skip = Opt
1239   , penalty        : integer = \c_max_int
1240   , after-penalty-vspace : skip = Opt
1241   , after-vspace   : skip = Opt
1242   , start-code     : tokenlist =      % no default values!
1243   , final-code     : tokenlist =      % no default values!
1244   , prefix         : tokenlist = \NoValue
1245   , heading-decls  : tokenlist = \normalfont
1246   , prefix-decls   : tokenlist =
1247   , number-decls   : tokenlist =
1248   , title-decls    : tokenlist =
1249   , subtitle-decls : tokenlist =
1250   , quote-decls    : tokenlist =
1251   , headformat-instance : tokenlist = std
1252   , number-format  : function(1) = \theheading
1253   , contents-extra : tokenlist =
1254 }
1255 \DeclareTemplateCode{heading}{display-v1}{9}
1256 {
1257   , name           = \l__head_name_tl
1258   , level          = \l__head_level_int
1259   , parent-name    = \l__head_pname_tl
1260   , reset-counter  = \l__head_reset_cnt_tl
1261   , placement      = {
1262     ,page = \__head_debug_typeout:n{ A~ page~ heading }
1263             \tl_set:Nn \l__head_placement_tl { page }
1264     ,top   = \__head_debug_typeout:n{ A~ top~ heading }
1265             \tl_set:Nn \l__head_placement_tl { top }
1266     ,normal = \__head_debug_typeout:n{ A~ normal~ heading }
1267             \tl_set:Nn \l__head_placement_tl { normal }
1268           }
1269   , mark-cmd       = \__head_mark_cmd:n
1270   , para-indent    = {
1271     ,true  = \@afterindenttrue
1272     ,false = \@afterindentfalse
1273   }

```

```

1274 , before-vspace = \l__head_before_skip
1275 , penalty       = \l__head_penalty_int
1276 , after-penalty-vspace = \l__head_after_penalty_skip
1277 , after-vspace  = \l__head_after_skip
1278 , start-code    = \l__head_start_code_tl
1279 , final-code    = \l__head_final_code_tl
1280 , prefix        = \l__head_typeset_prefix_tl
1281 , headformat-instance = \l__head_headformat_instance_tl
1282 , heading-decls = \l__head_heading_decls_tl
1283 , prefix-decls  = \l__head_prefix_decls_tl
1284 , number-decls  = \l__head_number_decls_tl
1285 , title-decls   = \l__head_title_decls_tl
1286 , subtitle-decls = \l__head_subtitle_decls_tl
1287 , quote-decls   = \l__head_quote_decls_tl
1288 , number-format = \__head_number_format:n
1289 , contents-extra = \l__head_contents_extra_tl
1290 }
1291 {
1292   \tl_set_eq:Nc \theheading { the \l__head_name_tl }
1293   \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}
1294   \__head_show_arguments:nnnnnnnnn
1295     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
1296   \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{display}{#1} }
1297   \tl_if_empty:oT \l__head_start_code_tl
1298   { \str_case:VnF \l__head_placement_tl
1299     {
1300       { page }
1301       { \tl_set:Nn \l__head_start_code_tl
1302         {
1303           \if@openright
1304             \cleardoublepage
1305           \else
1306             \clearpage
1307           \fi
1308           \thispagestyle{plain}
1309           \if@twocolumn
1310             \onecolumn
1311             \@tempswattrue
1312           \else
1313             \@tempswafalse
1314           \fi
1315           \null\vfil
1316         }
1317       }
1318       { top }
1319       { \tl_set:Nn \l__head_start_code_tl
1320         {
1321           \if@openright\cleardoublepage\else\clearpage\fi
1322           \thispagestyle{plain}
1323           \global\@topnum\z@
1324         }
1325       }
1326     }
1327   { \tl_clear:N \l__head_start_code_tl }

```

```

1328 }
1329 \tl_if_empty:oT \l__head_final_code_tl
1330 { \str_case:VnF \l__head_placement_tl
1331   {
1332     { page }
1333     { \tl_set:Nn \l__head_final_code_tl
1334       {
1335         \vfil\newpage
1336         \if@twoside
1337           \if@openright
1338             \null
1339             \thispagestyle{empty}
1340           \newpage
1341         \fi
1342       \fi
1343       \if@tempwa
1344         \twocolumn
1345       \fi
1346     }
1347   }
1348 }
1349 { \tl_set:Nn \l__head_final_code_tl { \@afterheading } }
1350 }
1351 \__head_determine_penalty:

```

Next lines are an inline version of `\__head_determine_number_typesetting:N` #2 This macro has changed so here we inline the original code;

```

1352 \bool_set:Nn \l__head_unnumbered_bool
1353 { \bool_lazy_or:p:nn
1354   { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
1355   { \bool_if_p:N #2 }
1356 }
1357 \bool_if:NTF \l__head_unnumbered_bool
1358 {
1359   \int_gset:Nn\c@secnumdepth{-99}
1360   \tl_clear:N \l__head_typeset_number_tl
1361 }
1362 {
1363   \@kernel@refstepcounter{ \l__head_name_tl }
1364   \protected@edef \l__head_typeset_number_tl
1365   {
1366     \__head_number_format:n { \l__head_name_tl }
1367   }
1368 }

```

End of inlined code.

```

1369 \__head_vertical_before_spacing:
1370 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
1371 \UseTaggingSocket{sec/begin}
1372   {{\int_use:N\l__head_level_int}
1373    {tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
1374 \__head_debug_typeout:n{use~ 'headformat'~instance:~
1375   \l__head_headformat_instance_tl }
1376 \use:e {

```

```

1377 \UseInstance{headformat} { \l__head_headformat_instance_tl }
1378 { \exp_not:o \UnusedTemplateKeys }
1379 { \exp_not:o { \l__head_typeset_prefix_tl } }
1380 { \exp_not:o { \l__head_typeset_number_tl } }
1381 { \exp_not:n { #3 } }
1382 { \exp_not:o { \use_i:nn #9 } }
1383 { \exp_not:o { \use_ii:nn #9 } }
1384 }
1385 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
1386 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
1387 \par \nobreak
1388 \skip_vertical:N \l__head_after_skip
1389 \l__head_final_code_tl
1390 \ignorespaces
1391 }

```

heading runin-v1 (*templ.*)

```

1392 \DeclareTemplateCopy{heading}{runin-v1}{display-v1}
1393 \DeclareTemplateCode{heading}{runin-v1}{9}
1394 {
1395   , name          = \l__head_name_tl
1396   , level         = \l__head_level_int
1397   , parent-name   = \l__head_pname_tl
1398   , reset-counter = \l__head_reset_cnt_tl
1399   , placement     = {
1400     page = \typeout{ ^^JA~ runin~ page~ placement~ heading-makes-no-sense
1401              (top-used)}
1402     \tl_set:Nn \l__head_placement_tl { top }
1403     ,top      = \__head_debug_typeout:n{ A~ top~ heading }
1404     \tl_set:Nn \l__head_placement_tl { top }
1405     ,normal   = \__head_debug_typeout:n{ A~ normal~ heading }
1406     \tl_set:Nn \l__head_placement_tl { normal }
1407   }
1408   , mark-cmd     = \__head_mark_cmd:n
1409   , para-indent  = {
1410     ,true  = %\typeout{para-indent~ setting~ ignored}
1411     ,false = %\typeout{para-indent~ setting~ ignored}
1412   }
1413   , before-vspace = \l__head_before_skip
1414   , penalty       = \l__head_penalty_int
1415   , after-penalty-vspace = \l__head_after_penalty_skip
1416   , after-vspace  = \l__head_after_skip
1417   , start-code    = \l__head_start_code_tl
1418   , final-code    = \l__head_final_code_tl
1419   , prefix        = \l__head_typeset_prefix_tl
1420   , headformat-instance = \l__head_headformat_instance_tl
1421   , heading-decls = \l__head_heading_decls_tl
1422   , prefix-decls  = \l__head_prefix_decls_tl
1423   , number-decls  = \l__head_number_decls_tl
1424   , title-decls   = \l__head_title_decls_tl
1425   , subtitle-decls = \l__head_subtitle_decls_tl
1426   , quote-decls   = \l__head_quote_decls_tl
1427   , number-format = \__head_number_format:n
1428   , contents-extra = \l__head_contents_extra_tl

```

```

1429 }
1430 {
1431   \__head_show_arguments:nnnnnnnnn
1432     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
1433   \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}
1434   \tl_set_eq:Nc \theheading { the \l__head_name_tl }
1435   \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{runin}{#1} }
1436   \tl_if_empty:oT \l__head_start_code_tl
1437   {
1438     \str_case:Vn \l__head_placement_tl
1439     {
1440       { top }    { \tl_set:Nn \l__head_start_code_tl {\clearpage} } }
1441     }
1442   }
1443   \__head_determine_penalty:

```

Next lines are an inline version of `\__head_determine_number_typesetting:N #2` This macro has changed so here we inline the original code;

```

1444   \bool_set:Nn \l__head_unnumbered_bool
1445     { \bool_lazy_or_p:nn
1446       { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
1447       { \bool_if_p:N #2 }
1448     }
1449   \bool_if:NTF \l__head_unnumbered_bool
1450   {
1451     \int_gset:Nn\c@secnumdepth{-99}
1452     \tl_clear:N \l__head_typeset_number_tl
1453   }
1454   {
1455     \@kernel@refstepcounter{ \l__head_name_tl }
1456     \protected@edef \l__head_typeset_number_tl
1457       {
1458         \__head_number_format:n { \l__head_name_tl }
1459       }
1460   }

```

End of inlined code.

```

1461   \__head_vertical_before_spacing:
1462   \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
1463   \UseTaggingSocket{sec/begin}
1464     {{\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
1465   \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
1466   \def \@svsechd {
1467     \__head_debug_typeout:n{use~ 'headformat'~instance:~
1468       \l__head_headformat_instance_tl }
1469     \use:e {
1470       \UseInstance{headformat} { \l__head_headformat_instance_tl }
1471       { \exp_not:o \UnusedTemplateKeys }
1472       { \exp_not:o { \l__head_typeset_prefix_tl } }
1473       { \exp_not:o { \l__head_typeset_number_tl } }
1474       { \exp_not:n { #3 } }
1475       { \exp_not:o { \use_i:nn #9 } }
1476       { \exp_not:o { \use_ii:nn #9 } }
1477     }

```

```

1478     \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
1479     \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
1480   }
1481   \@nobreakfalse
1482   \global\@noskipsectrue
1483   \everypar{%
1484     \if@noskipsec
1485       \global\@noskipsecfalse
1486       {\setbox\z@\lastbox}
1487       \clubpenalty\@M
1488       \@svsechd
1489       \unskip
1490       \UseTaggingSocket{sec/title/split}
1491       \skip_horizontal:N \l__head_after_skip
1492     \else
1493       \clubpenalty \@clubpenalty
1494       \everypar{}%
1495     \fi
1496   }
1497   \l__head_final_code_tl
1498   \ignorespaces
1499 }

```

headformat display-v1 (*templ.*) The display template produces

```

1500 \DeclareTemplateInterface{headformat}{display-v1}{6}
1501 {
1502   , heading-indent      : length = Opt
1503   , title-format        : function(1) = #1
1504   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
1505   , number-title-sep    : tokenlist = 20pt
1506 }

```

headformat hang-v1 (*templ.*)

```

1507 \DeclareTemplateInterface{headformat}{hang-v1}{6}
1508 {
1509   , heading-indent      : length = Opt
1510   , title-format        : function(1) = #1
1511   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
1512   , number-title-sep    : tokenlist = 1em
1513 }

```

headformat runin-v1 (*templ.*)

```

1514 \DeclareTemplateInterface{headformat}{runin-v1}{6}
1515 {
1516   , heading-indent      : length = Opt
1517   , title-format        : function(1) = #1
1518   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
1519   , number-title-sep    : tokenlist = 1em
1520 }

```

headformat display-v1 (*templ.*)

```

1521 \DeclareTemplateCode{headformat}{display-v1}{6}
1522   % args: keys,prefix,number,title,subtitle,quotation

```

```

1523 {
1524 , heading-indent      = \l__head_heading_indent_dim
1525 , title-format        = \__head_title_format:n
1526 , prefix-number-sep   = \l__head_prefix_number_sep_tl
1527 , number-title-sep    = \l__head_number_title_sep_tl
1528 }
1529 {
1530 \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}
1531 \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{display-v1}{#1} }
1532 \group_begin:
1533   \UseTaggingSocket{sec/title/begin}{\int_use:N\l__head_level_int}{#4}}
1534   \normalfont
1535   \interlinepenalty \@M
1536   \l__head_heading_decls_tl{}
1537   \bool_if:NTF \l__head_unnumbered_bool
1538   {
1539     \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1540     {
1541       \skip_horizontal:N \l__head_heading_indent_dim
1542       \MakeLinkTarget[\l__head_name_tl]{}
1543     }
1544     {
1545       \MakeLinkTarget[\l__head_name_tl]{}
1546       \skip_horizontal:N \l__head_heading_indent_dim
1547     }
1548   }
1549   {
1550     \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1551     {
1552       \skip_horizontal:N \l__head_heading_indent_dim
1553       \MakeLinkTarget{\l__head_name_tl}
1554     }
1555     {
1556       \MakeLinkTarget{\l__head_name_tl}
1557       \skip_horizontal:N \l__head_heading_indent_dim
1558     }
1559     \IfNoValueF{#2}
1560     {
1561       { \l__head_prefix_decls_tl #2 }
1562       \nobreak
1563       \skip_horizontal:n { \l__head_prefix_number_sep_tl }
1564     }
1565     \l__head_number_decls_tl
1566     #3
1567     \par\nobreak
1568     \skip_vertical:n { \l__head_number_title_sep_tl }
1569   }
1570   \l__head_title_decls_tl
1571   \__head_title_format:n {#4}
1572   \par
1573   \UseTaggingSocket{sec/title/end}
1574 \group_end:
1575 }

```

headformat hang-v1 (*templ.*)

```

1576 \DeclareTemplateCode{headformat}{hang-v1}{6}
1577 {
1578   , heading-indent      = \l__head_heading_indent_dim
1579   , title-format        = \__head_title_format:n
1580   , prefix-number-sep   = \l__head_prefix_number_sep_tl
1581   , number-title-sep    = \l__head_number_title_sep_tl
1582 }
1583 {
1584   \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}
1585   \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang-v1}{#1} }
1586   \group_begin:
1587     \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
1588     \normalfont
1589     \interlinepenalty \@M
1590     \l__head_heading_decls_tl{}
1591     \bool_if:NTF \l__head_unnumbered_bool
1592     {
1593       \tl_set:Nn\l__head_tmpa_tl
1594       {
1595         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1596         {
1597           \skip_horizontal:N \l__head_heading_indent_dim
1598           \MakeLinkTarget[\l__head_name_tl]{}
1599         }
1600         {
1601           \MakeLinkTarget[\l__head_name_tl]{}
1602           \skip_horizontal:N \l__head_heading_indent_dim
1603         }
1604       }
1605     }
1606     {
1607       \tl_set:Nn \l__head_tmpa_tl
1608       {
1609         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1610         {
1611           \skip_horizontal:N \l__head_heading_indent_dim
1612           \MakeLinkTarget{\l__head_name_tl}
1613         }
1614         {
1615           \MakeLinkTarget{\l__head_name_tl}
1616           \skip_horizontal:N \l__head_heading_indent_dim
1617         }
1618       }
1619       \IfNoValueF{#2}
1620       {
1621         { \l__head_prefix_decls_tl #2 }
1622         \nobreak
1623         \skip_horizontal:n { \l__head_prefix_number_sep_tl }
1624       }
1625       { \l__head_number_decls_tl #3 }
1626       \skip_horizontal:n { \l__head_number_title_sep_tl }
1627     }
1628     \UseTaggingSocket{sec/title/hang}

```

```

1629     {{\int_use:N\l__head_level_int}\l__head_unnumbered_bool{\l__head_tmpa_tl}{#4}}
1630     {
1631         \@hangfrom
1632         {
1633             \l__head_tmpa_tl
1634         }
1635     }
1636     \l__head_title_decls_tl
1637     \__head_title_format:n {#4}
1638     \par
1639     \group_end:
1640 }

```

headformat runin-v1 (*templ.*)

```

1641 \DeclareTemplateCode{headformat}{runin-v1}{6}
1642 {
1643     , heading-indent      = \l__head_heading_indent_dim
1644     , title-format       = \__head_title_format:n
1645     , prefix-number-sep = \l__head_prefix_number_sep_tl
1646     , number-title-sep  = \l__head_number_title_sep_tl
1647 }
1648 {
1649     \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}
1650     \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{runin-v1}{#1} }
1651     \group_begin:
1652         \normalfont
1653         \interlinepenalty \@M
1654         \l__head_heading_decls_tl{}
1655         \bool_if:NTF \l__head_unnumbered_bool
1656         {
1657             \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1658             {
1659                 \skip_horizontal:N \l__head_heading_indent_dim
1660                 \MakeLinkTarget[\l__head_name_tl]{}
1661             }
1662             {
1663                 \MakeLinkTarget[\l__head_name_tl]{}
1664                 \skip_horizontal:N \l__head_heading_indent_dim
1665             }
1666         }
1667         {
1668             \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
1669             {
1670                 \skip_horizontal:N \l__head_heading_indent_dim
1671                 \MakeLinkTarget{\l__head_name_tl}
1672             }
1673             {
1674                 \MakeLinkTarget{\l__head_name_tl}
1675                 \skip_horizontal:N \l__head_heading_indent_dim
1676             }
1677             {
1678                 \UseTaggingSocket{sec/title/number}{\int_use:N\l__head_level_int}
1679                 {
1680                     \IfNoValueF{#2}

```

```

1681         {
1682             { \l__head_prefix_decls_tl #2 }
1683             \nobreak
1684             \skip_horizontal:n { \l__head_prefix_number_sep_tl }
1685         }
1686         \l__head_number_decls_tl
1687         #3
1688     }
1689 }
1690 \skip_horizontal:n { \l__head_number_title_sep_tl }
1691 }
1692 \l__head_title_decls_tl
1693 \__head_title_format:n {#4}
1694 \group_end:
1695 }

```

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	
\!	151, 153
\,	152, 154
\.	151, 153
\:	152, 154
\;	152, 154
\?	151, 153
\@startsection	19
_	249, 260, 271
_	16, 17
A	
\addcontents	14, 15, 61
\addcontentsline	20, 47, 885
\addcontentslinebookmark	848, 874
\addcontentslinebookmarkOff	848, 884
\addcontentslinebookmarkOn	848
\addcontentslinebookmarkReset	850, 858, 861, 893
\addpenalty	33, 35–37, 46, 454, 458, 459
\addpenaltywithfil	451, 459
\AddPunct	156
\addtocontents	926, 927
\AddToHook	67, 494, 852, 1109, 1159, 1184, 1194
\advspace	37, 834, 926, 927
\AssignStructureRole	687
\AssignTaggingSocketPlug	637, 638, 639, 707, 708, 709, 768, 769, 770
\AtBeginDocument	929
B	
\baselineskip	13, 14
\bfseries	907, 922, 938, 949, 959, 969, 981, 1170
bool commands:	
\bool_gset_false:N	18
\bool_gset_true:N	13
\bool_if:NTF	24, 26, 95, 330, 410, 424, 439, 556, 649, 678, 683, 721, 779, 807, 814, 841, 876, 887, 1357, 1449, 1537, 1591, 1655
\bool_if_p:N	811, 1355, 1447
\bool_lazy_and_p:nn	1087, 1094
\bool_lazy_any:nTF	1080
\bool_lazy_or_p:nn	809, 1353, 1445
\bool_new:N	8, 59
\bool_set:Nn	808, 1352, 1444
\bool_set_false:N	83, 339, 559
\bool_set_true:N	78, 688
\BooleanFalse	10, 95
\BooleanTrue	10, 95, 1205, 1208, 1209
\box	744
C	
\centering	907
\chapter	4, 5, 9, 16–19, 48–50, 55, 64, 1139, 1140, 1143, 1188, 1198
\chaptermark	12, 60, 925
\cleardoublepage	348, 364, 1304, 1321

<code>\clearpage</code>	12, 46, 59, 350, 364, 564, 1306, 1321, 1440
<code>\clubpenalty</code>	610, 616, 1487, 1493
cmd commands:	
<code>\cmd_arg_spec:N</code>	1075
<code>\cs</code>	443
cs commands:	
<code>\cs_generate_variant:Nn</code>	1107
<code>\cs_gset_protected:Npe</code>	23, 25
<code>\cs_if_exist:NTF</code>	930
<code>\cs_if_exist_p:N</code>	1088, 1095
<code>\cs_new:Npn</code> 31, 40, 123, 136, 802, 806, 825, 866, 930, 1060, 1066, 1071, 1079	
<code>\cs_new_eq:NN</code>	9, 10
<code>\cs_new_protected:Npn</code> . 11, 16, 21, 28, 29, 68, 155, 156, 449, 459, 491, 1215	
<code>\cs_parameter_spec:N</code>	1082, 1090, 1097, 1107
<code>\cs_set_eq:NN</code>	370, 452, 456, 566
<code>\cs_set_protected:Npn</code>	488
<code>\cs_to_str:N</code>	1088, 1090, 1113
<code>\csname</code>	497
D	
<code>\DebugHeadingsOff</code>	20, 28
<code>\DebugHeadingsOn</code>	20, 28
<code>\DeclareDocumentCommand</code>	1013, 1127, 1143, 1180, 1186, 1188, 1196, 1198, 1217, 1219, 1221, 1223, 1225
<code>\DeclareInstance</code>	. 58, 899, 912, 931, 941, 952, 962, 974, 986, 993, 1000, 1004, 1008, 1029, 1041, 1045, 1055, 1161, 1174
<code>\DeclareInstanceCopy</code> .	992, 997, 998, 999
<code>\DeclareRobustCommand</code>	54
<code>\DeclareTemplateCode</code> 29, 274, 500, 623, 691, 753, 1255, 1393, 1521, 1576, 1641	
<code>\DeclareTemplateCopy</code>	29, 1392
<code>\DeclareTemplateInterface</code> . 168, 204, 240, 252, 263, 1229, 1500, 1507, 1514	
<code>\def</code>	54, 151, 153, 589, 1466
dim commands:	
<code>\dim_compare:nNnTF</code>	413, 466, 470, 472, 480, 650, 658, 722, 730, 780, 788, 835, 843, 1539, 1550, 1595, 1609, 1657, 1668
<code>\dim_set:Nn</code>	469
<code>\l_tmpa_dim</code>	469, 477, 480, 481
<code>\c_zero_dim</code>	466, 473, 480
<code>\DocumentMetadata</code>	3, 26
E	
<code>\EditTemplateDefaults</code>	29
<code>\else</code>	160, 338, 349, 356, 364, 496, 615, 831, 860, 1035, 1042, 1202, 1305, 1312, 1321, 1492
<code>\endcsname</code>	495, 497
<code>\everypar</code> ...	40, 606, 617, 830, 1483, 1494
exp commands:	
<code>\exp_args:Ne</code>	1062, 1073
<code>\exp_not:N</code>	93, 1032, 1038, 1048
<code>\exp_not:n</code>	33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47, 48, 49, 50, 71, 72, 94, 96, 97, 98, 99, 100, 101, 102, 103, 134, 411, 418, 419, 420, 421, 422, 423, 594, 595, 596, 597, 598, 599, 896, 1036, 1037, 1051, 1052, 1121, 1378, 1379, 1380, 1381, 1382, 1383, 1471, 1472, 1473, 1474, 1475, 1476
<code>\expandafter</code>	497
<code>\ExpandArgs</code>	1120, 1135
F	
<code>\fbox</code>	4
<code>\fi</code>	163, 164, 340, 351, 358, 364, 388, 389, 392, 498, 618, 827, 837, 862, 1025, 1035, 1056, 1211, 1307, 1314, 1321, 1341, 1342, 1345, 1495
<code>\filbreak</code>	35, 46
<code>\font</code>	146, 147, 148
<code>\fontdimen</code>	146, 147, 148
<code>\frenchspacing</code>	26, 27, 151
G	
<code>\global</code> ...	366, 605, 608, 1323, 1482, 1485
<code>\glueexpr</code>	145
group commands:	
<code>\group_begin:</code>	464, 636, 705, 766, 1532, 1586, 1651
<code>\group_end:</code>	484, 674, 750, 800, 1574, 1639, 1694
H	
<code>\hangindent</code>	743
<code>\hbox</code>	720
head commands:	
<code>\head_debug_off:</code>	20, 11, 16, 29
<code>\head_debug_on:</code>	20, 11, 11, 28
head internal commands:	
<code>\l_head_after_penalty_skip</code>	300, 413, 414, 527, 835, 836, 843, 844, 1276, 1415
<code>\l_head_after_skip</code>	301, 426, 441, 528, 614, 1277, 1388, 1416, 1491
<code>\l_head_before_skip</code>	298, 525, 834, 1274, 1413

<code>\l__head_bookmark_tl</code>	53, 77, 82, 99, 109	<code>__head_mark_cmd:n</code>	293, 520, 870, 1269, 1408
<code>\l__head_column_spanning_bool</code>	34, 292, 330, 339, 410, 424, 439, 519, 556, 559, 841	<code>\l__head_name_tl</code>	276, 325, 502, 554, 652, 654, 660, 662, 724, 726, 732, 734, 782, 784, 790, 792, 821, 874, 878, 885, 889, 1257, 1292, 1363, 1366, 1395, 1434, 1455, 1458, 1542, 1545, 1553, 1556, 1598, 1601, 1612, 1615, 1660, 1663, 1671, 1674
<code>\l__head_contents_extra_tl</code>	322, 549, 895, 1289, 1428	<code>\l__head_nameref_tl</code>	56, 85, 100, 112
<code>__head_debug:n</code>	9, 9, 23	<code>\l__head_number_decls_tl</code>	309, 536, 1284, 1423, 1565, 1624, 1686
<code>\g__head_debug_bool</code>	8, 13, 18, 24, 26	<code>__head_number_format:n</code>	315, 542, 1288, 1366, 1427, 1458
<code>__head_debug_gset:</code>	11, 14, 19, 21	<code>\l__head_number_tag_tl</code>	321, 548
<code>__head_debug_typeof:n</code>	9, 10, 25, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 69, 70, 91, 125, 129, 134, 139, 159, 161, 283, 285, 287, 289, 334, 406, 450, 512, 514, 516, 557, 590, 896, 1015, 1262, 1264, 1266, 1374, 1403, 1405, 1467	<code>\l__head_number_title_sep_tl</code>	1527, 1568, 1581, 1625, 1646, 1690
<code>__head_determine_number_typesetting:N</code>	66, 68, 399, 581, 806, 806	<code>\l__head_number_tl</code>	45, 46, 60, 420, 596, 643, 713, 774, 816, 822
<code>__head_determine_penalty:</code>	398, 580, 802, 802, 1351, 1443	<code>\l__head_order_clist</code>	627, 696, 757
<code>\l__head_final_code_tl</code>	31, 33, 39, 303, 377, 381, 396, 445, 530, 573, 576, 578, 620, 1279, 1329, 1333, 1349, 1389, 1418, 1497	<code>\l__head_penalty_int</code>	299, 526, 803, 804, 833, 1275, 1414
<code>__head_find_label:w</code>	73, 123, 123	<code>\l__head_placement_tl</code>	60, 284, 286, 288, 290, 332, 336, 343, 378, 511, 513, 515, 517, 562, 574, 1263, 1265, 1267, 1298, 1330, 1402, 1404, 1406, 1438
<code>__head_find_label_aux:w</code>	25, 132, 136, 141	<code>\l__head_pname_tl</code>	279, 505, 1259, 1397
<code>__head_handle_marks_etc:nnnn</code>	434, 601, 866, 866, 1385, 1478	<code>\l__head_prefix_decls_tl</code>	308, 535, 1283, 1422, 1561, 1620, 1682
<code>\l__head_headformat_instance_tl</code>	306, 407, 417, 533, 591, 593, 1281, 1375, 1377, 1420, 1468, 1470	<code>__head_prefix_format:n</code>	314, 541
<code>\l__head_heading_decls_tl</code>	307, 534, 648, 718, 778, 1018, 1036, 1051, 1282, 1421, 1536, 1590, 1654	<code>\l__head_prefix_number_sep_tl</code>	1526, 1563, 1580, 1622, 1645, 1684
<code>__head_heading_format:n</code>	313, 540, 667	<code>\l__head_prefix_tl</code>	45, 304, 419, 531, 595, 642, 712, 773, 817
<code>\l__head_heading_indent_dim</code>	626, 650, 651, 655, 658, 659, 663, 694, 722, 723, 727, 730, 731, 735, 756, 780, 781, 785, 788, 789, 793, 1524, 1539, 1541, 1546, 1550, 1552, 1557, 1578, 1595, 1597, 1602, 1609, 1611, 1616, 1643, 1657, 1659, 1664, 1668, 1670, 1675	<code>__head_prep_decls_and_auxi:nnNN</code>	1063, 1066
<code>\l__head_instance_keys_tl</code>	24, 60, 89, 90, 94	<code>__head_prep_decls_and_auxii:nNwNN</code>	1068, 1071
<code>\l__head_label_tl</code>	23, 25, 60, 101, 121, 126, 130, 140	<code>__head_prep_decls_and_auxiii:nnNnNN</code>	1074, 1079
<code>\l__head_level_int</code>	277, 401, 403, 404, 408, 503, 583, 585, 586, 588, 810, 1258, 1354, 1370, 1372, 1373, 1396, 1446, 1462, 1464, 1465, 1533, 1587, 1629, 1678	<code>__head_prep_decls_and_format_from_tl:nNN</code>	1017, 1060, 1060
		<code>__head_process_shorttitle:nN</code>	1119, 1134, 1157
		<code>__head_punct_format:n</code>	316, 543
		<code>\l__head_punct_tl</code>	45, 305, 532, 818
		<code>\l__head_quote_decls_tl</code>	312, 539, 1287, 1426
		<code>__head_quote_format:n</code>	319, 546
		<code>\l__head_quote_tl</code>	58, 87, 103, 120
		<code>\l__head_reset_cnt_tl</code>	280, 506, 1260, 1398
		<code>\l__head_running_tl</code>	54, 76, 81, 98, 110

Q

quark commands:
`\q_no_value` 25, 26, 73
`\quark_if_no_value:nTF` 124, 138
`\q_stop` 1068, 1071

R

`\raggedright` 922, 1170
`\refstepcounter` 20, 46
`\relax` 149, 157, 861, 1020, 1026, 1117, 1121
`\renewcommand` 856
`\RenewCommandCopy` 854, 1110
`\RenewDocumentCommand` 1111
`\RestoreLastSkip` 13, 14, 41, 60

S

`\SaveLastSkip` 13, 14, 41, 60
scan commands:
`\scan_stop:` 489
sec commands:
`\sec_add_penalty_with_fil:n`
..... 35, 36, 46, 371, 449, 449, 567
`\sec_add_penalty_with_possible_-
fil:n` 33,
35, 36, 46, 370, 449, 453, 457, 566, 832
`\secdef` 19
`\secdef` 4, 5, 9, 19, 48, 50, 55, 1111
`\section` .. 5, 8, 9, 17, 18, 48, 52, 62, 1217
`\sectionmark` 12, 60, 935
`\setbox` 609, 720, 1486
`\SetCurrentTemplateKeys` 32, 58
`\SetKnownTemplateKeys` 329, 555
`\SetTemplateKey` 32
`\SetTemplateKeys` 635,
704, 765, 1296, 1435, 1531, 1585, 1650
`\sfcode` 27, 151, 152, 153, 154
skip commands:
`\skip_add:Nn` 477
`\skip_horizontal:N`
. 614, 651, 655, 659, 663, 723, 727,
731, 735, 781, 785, 789, 793, 1491,
1541, 1546, 1552, 1557, 1597, 1602,
1611, 1616, 1659, 1664, 1670, 1675
`\skip_horizontal:n`
..... 1563, 1622, 1625, 1684, 1690
`\skip_set_eq:NN` 465, 468
`\skip_vertical:N` 426, 441, 1388
`\skip_vertical:n` 1568
`\l_tmpa_skip` 465, 466, 468, 482
`\l_tmpb_skip` 36, 468, 477, 478
`\c_zero_skip` 650, 658, 722, 730, 780,
788, 1539, 1550, 1595, 1609, 1657, 1668
`\spacefactor` 26, 27, 155, 158
`\specialsection` 52

str commands:

`\c_hash_str` 1082, 1091, 1098
`\str_case:nn` 562, 1438
`\str_case:nnTF`
..... 343, 378, 574, 1113, 1298, 1330
`\str_if_eq:nnTF` 332
`\str_if_eq_p:nn` 1082,
1083, 1084, 1085, 1086, 1089, 1096
`\string` 1016
`\subparagraph` 17, 18, 63, 1225
`\subparagraphmark` 982
`\subsection` 17, 18, 62, 1219
`\subsectionmark` 946
`\subsubsection` 17, 18, 62, 1221

T

tag commands:
`\tag_if_active:TF` 1148
`\tag_mc_end_push:` 677, 682
`\tag_struct_begin:n` 679, 684
`\tag_struct_end:` 678, 683
tag internal commands:
`\l__tag_para_bool` 678, 683
`\l__tag_para_flattened_bool` 688
`\tagpdfparaOff` 640, 710, 771
`\tagpdfparaOn` 689
template commands:
`\template_process_order_clist:nnn`
..... 668, 737, 745, 796
template types:
`headformat` 167
`heading` 166
templates:
`headformat display` 240, 623
`headformat display-v1` 1500, 1521
`headformat hang` 252, 691
`headformat hang-v1` 1507, 1576
`headformat runin` 263, 753
`headformat runin-v1` 1514, 1641
`heading display` 168, 274
`heading display-v1` 1229
`heading runin` 204, 500
`heading runin-v1` 1392
T_EX and L^AT_EX 2_ε commands:
`\@` 27
`\@LRmoderr` 461
`\@M` 610,
647, 717, 777, 1487, 1535, 1589, 1653
`\@addpunct` 27
`\@afterheading` 396, 1349
`\@afterindentfalse` .. 296, 1024, 1272
`\@afterindenttrue` ... 295, 1021, 1271
`\@chapapp` 50, 920, 929
`\@chappap` 28

<code>\UseStructureName</code>	200,	<code>\vfilneg</code>	33, 35, 37, 492
	236, 404, 586, 679, 684, 1373, 1464	<code>\vskip</code>	478, 481, 482
<code>\UseTaggingSocket</code>		<code>\vspace</code> . .	16, 248, 414, 836, 844, 990, 1178
	. 401, 402, 408, 430, 583, 584, 588,	<code>\vspace*</code>	46
	613, 1370, 1371, 1462, 1463, 1465,		
	1490, 1533, 1573, 1587, 1628, 1678		
		W	
	V	<code>\wd</code>	743
<code>\vfil</code> .	33, 35, 37, 359, 382, 492, 1315, 1335	<code>\WordSpaceAmount</code> . . .	144, 1504, 1511, 1518