

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

$\text{\LaTeX}$ Project*

v0.9k 2026-08-06

Abstract

Contents

1	Introduction	2
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	4
2.5	Converting heading commands defined with <code>\@startsection</code>	5
2.5.1	Step 1: Getting the template name and the instance values	5
2.5.2	Step 2: Get the default template values	6
2.6	Step 3: Declare the instances	7
2.6.1	Step 4 Redefine the <code>\section</code> command	7
3	Setting up $\text{\LaTeX}$ 2ϵ heading commands in classes	7
4	Open points	8
5	Template types and templates for headings	8
5.1	Template types	8
5.1.1	The template type ‘heading’	8
5.1.2	The template type ‘headformat’	9
5.2	Templates	10
5.2.1	Templates of type <code>heading</code>	10
5.2.2	The <code>heading</code> template ‘ <code>\all</code> ’	10
5.2.3	The <code>heading</code> template ‘display’	11
5.2.4	The <code>heading</code> template ‘runin’	12
5.2.5	Templates of type <code>headformat</code>	13
5.2.6	The <code>headformat</code> template ‘hang’	13
5.2.7	The <code>headformat</code> template ‘runin’	14
5.2.8	The <code>headformat</code> template ‘display’	14
5.3	Default instances	15

*Initial implementation by Frank Mittelbach.

5.3.1	Instances of <code>heading</code> templates	15
5.3.2	Instances of <code>headformat</code> templates	15
6	Support for legacy classes and packages	16
6.1	Support classes based on legacy $\text{\LaTeX} 2_{\epsilon}$ interfaces	16
6.2	Support for the <code>titlesec</code> package interfaces	16
7	Support for links	16
8	Bookmarks	17
9	Tagging support	17
10	Debugging support	17
11	The Implementation	17
11.1	Debugging	17
11.2	Temp variable(s)	19
11.3	variable(s)	19
11.4	New user commands	19
11.5	The $\text{\LaTeX} 2_{\epsilon}$ parsing interface	20
11.6	General helper commands	23
11.7	Templates	24
11.7.1	Template types	24
11.7.2	heading templates interfaces	25
11.7.3	headformat templates interfaces	26
11.7.4	heading templates code	26
11.7.5	headformat templates code	36
11.7.6	Internal commands used by the template code	40
11.8	Support for legacy classes and packages	43
11.8.1	Instances (sample/default definitions)	44
11.8.2	New <code>\@startsection</code>	47
11.8.3	New <code>\secdef</code>	48
11.8.4	Core $\text{\LaTeX}$	49
12	Issues and problems noticed along the way	51
12.1	Local <code>\DeclareInstance</code>	51
12.2	<code>\SetCurrentTemplateKeys</code>	51
12.3	O-expansion of <code>\UseInstance</code> arguments	52
12.4	Switch <code>\openright</code> is not defined by core	52
12.5	Indexheading at least for <code>l3doc</code> is wrong now	52
	Index	52

1 Introduction

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

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

2 Changes to the user interface

2.1 New user commands

`\theheading` This command expands inside a heading to the current number.
`\partmark` This replaces the fix `\markboth{}{}` or similar in the standard `\part` definition.

2.2 The optional argument

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

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

The star-form `\section*{Title}` stops the numbering, toc, bookmark and running header as it was the way for the last 30-odd years. New is that the optional argument is handled as a key/val list if an equal sign at the outer level is detected, otherwise the argument is taken to be the value of the 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 restrictions by converting such sectioning commands to use the new interfaces.

2.5 Converting heading commands defined with `\@startsection`

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

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

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

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

\begin{document}

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

\end{document}
```

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

```
The instance 'section-@startsection' of type 'heading' has values:
> level => 1
> placement => normal
> mark-cmd => \sectionmark {##1}
> para-indent => false
```

```

> before-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 =>
> heading-decls => \tubsecfmt
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> headformat-instance => section-@startsection
> number-format => \theheading
> contents-extra =>
> name => section
> from template => display.

```

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

```

> heading-indent => 0pt
> title-format => \tubsecfmt {##1}
> prefix-number-sep => 0pt
> number-title-sep => 1em
> from template => hang.

```

2.5.2 Step 2: Get the default template values

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

```

\DocumentMetadata{}
\documentclass{ltugboat}

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

```

This gives in the log and terminal:

The template 'display' of type 'heading' has default values:

```

> level => 0
> placement => normal
> mark-cmd =>
> para-indent => false
> before-vspace => 0pt
> penalty => \c_max_int
> after-penalty-vspace => 0pt
> after-vspace => 0pt

```

```

> start-code =>
> final-code =>
> heading-decls => \normalfont
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> headformat-instance => std
> number-format => \theheading
> contents-extra => .

```

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

```

> heading-indent => 0pt
> title-format => ##1
> prefix-number-sep => 0pt
> number-title-sep => 1em.

```

2.6 Step 3: Declare the instances

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

This leads then to these declarations (note that in the `title-format ##1` should be changed into `#1`):

```

\DeclareInstance{heading}{section}{display} % #1=heading, #2=name, #3=template name
{
  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
}
\DeclareInstance{headformat}{section}{hang} % #1=heading, #2=name, #3=template name
{
  title-format   = \tubsecfmt {#1}
}

```

2.6.1 Step 4 Redefine the \section command

Finally, the heading should be defined to use the new interface.

```

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

```

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

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

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

The name of the `heading` instance is given in the first argument of `\ParseLaTeXeHeading` and it is recommended that classes use the same name as the heading command, to make it easier for users to identify the instance to edit if they want to adapt the command. Examples of instances that mimic the behavior of the heading commands in the standard classes can be found below. Section 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 necessary data arguments all in the last positional argument is actually better (besides being faster).

5.1.1 The template type ‘heading’

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

Arg: 2 unnumbered heading?

Arg: 3 main title of the heading

Arg: 4 toc title

Arg: 5 running title

Arg: 6 bookmark title

Arg: 7 nameref text

Arg: 8 label for the heading in the form `\label{<string>}`

Arg: 9 { sub title } { quotation }

Semantics:

Handles the layout and processing aspects of a heading.

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

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

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

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

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

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

5.1.2 The template type ‘headformat’

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

Arg: 2 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, subtitle or quotation then this is indicated with `\NoValue`.

The template can ignore the `\fixed prefix text` in certain circumstance (or even always). The templates currently implemented to mimic $\text{\LaTeX} 2_{\epsilon}$ behavior do so in case of unnumbered headings. Same for `\subtitle` and `\quotation`: both are always ignored by the currently defined templates.

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

5.2 Templates

5.2.1 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.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`

¹Maybe questionable

- 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**
- 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 pagebreak 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`
- 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: **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 (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>`

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!

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 (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!

5.2.5 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. There is now the new order key mechanism to make this happen so the templates are going to change soon (and will different keys and some additional ones)!

5.2.6 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

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!

5.2.7 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

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 $\text{\TeX}$ 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.

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 L^AT_EX always used by default for `\paragraph` and `\subparagraph`.

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

5.2.8 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

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 T_EX 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 T_EX 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!

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`.

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}{before skip}{after skip}{style}
```

with a special logic that the sign of `<before skip>` determines display or runin heading and that of `<after skip>` 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]
```

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{\<counter>}` while unnumbered heading commands use `\MakeLinkTarget[\<counter>]{}`.

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

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

8 Bookmarks

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

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 {\ltlabsecIIDate}
6 {\ltlabsecIIversion}
7 {heading implementation}

```

11.1 Debugging

```

\g__head_debug_bool

8 \bool_new:N \g__head_debug_bool

(End of definition for \g__head_debug_bool.)

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

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

\head_debug_on:
\head_debug_off:
\__head_debug_gset:
11 \cs_new_protected:Npn \head_debug_on:
12 {
13   \bool_gset_true:N \g__head_debug_bool
14   \__head_debug_gset:
15 }

16 \cs_new_protected:Npn \head_debug_off:
17 {
18   \bool_gset_false:N \g__head_debug_bool
19   \__head_debug_gset:
20 }

21 \cs_new_protected:Npn \__head_debug_gset:
22 {
23   \cs_gset_protected: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 17.)

\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 17.)

`\__head_show_arguments:nnnnnn`

Some debug data ...

```

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:nnnnnnnn`

Some debug data ...

```

40 \cs_new:Npn \__head_show_arguments:nnnnnnnn #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:nnnnnnnn`.)

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

```

But this token lists needs a declaration:

```

60 \tl_new:N\l__head_label_tl
61 \tl_new:N\l__head_instance_keys_tl

```

```

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

```

```

62 \tl_new:N\l__head_typeset_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 {
76   \tl_clear:N \l__head_toc_tl
77   \tl_clear:N \l__head_running_tl
78   \tl_clear:N \l__head_bookmark_tl
79   \bool_set_true:N \l__head_unnumbered_bool
80 }
81 {
82   \tl_set_eq:NN \l__head_toc_tl \l__head_title_tl
83   \tl_set_eq:NN \l__head_running_tl \l__head_title_tl
84   \tl_set_eq:NN \l__head_bookmark_tl \l__head_title_tl
85   \bool_set_false:N \l__head_unnumbered_bool
86 }

```

The nameref always starts out matching the main title.

```

87 \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

```

88 \tl_set_eq:NN \l__head_subtitle_tl \c_novalue_tl
89 \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?

```

90 \IfNoValueTF { #3 }
91 { \tl_clear:N \l__head_instance_keys_tl }
92 { \keys_set_known:nnN {heading} { #3 } \l__head_instance_keys_tl }

```

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

```

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

```

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

```

103         { \exp_not:o \l__head_label_tl }
104         { { \exp_not:o \l__head_subtitle_tl }
105           { \exp_not:o \l__head_quote_tl } }
106     }
107 }

```

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.

```

108 \keys_define:nn {heading} {
109   , shorttitle .meta:n =
110     {toc = {#1} , running = {#1} , bookmark = {#1} , nameref= {#1} }
111   , bookmark .tl_set:N = \l__head_bookmark_tl
112   , running .tl_set:N = \l__head_running_tl
113   , toc .tl_set:N = \l__head_toc_tl
114   , nameref .tl_set:N = \l__head_nameref_tl
115   %
116   , numbered .bool_set_inverse:N = \l__head_unnumbered_bool
117   , numbered .default:n = true
118   , unnumbered .bool_set:N = \l__head_unnumbered_bool
119   , unnumbered .default:n = true
120   %
121   , subtitle .tl_set:N = \l__head_subtitle_tl
122   , 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.

```

123   , label .code:n = \tl_put_right:Nn \l__head_label_tl { \label{#1} }
124 }

```

(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.

```

125 \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.

```

126   \quark_if_no_value:nTF {#2}
127   {
128     \__head_debug_typeout:n{---no-label }
129     \tl_clear:N \l__head_label_tl
130     \tl_set:Nn\l__head_title_tl {#1#3}
131   }

```

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

```

132   {
133     \__head_debug_typeout:n{---label~found~#2}
134     \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`.

```

135     \tl_set:Nn\l__head_title_tl {#1}
136     \__head_find_label_aux:w #3\__head_find_label_aux:w
137   }
138   \__head_debug_typeout:n{---~return:~ '\exp_not:o \l__head_title_tl' }
139 }

```

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.

```

140 \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.

```

141   \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.

```

142   \quark_if_no_value:nF {#2}
143   {
144     \__head_debug_typeout:n{---~ extra~ label~ '#2'~ found }
145     \tl_put_right:Nn \l__head_label_tl { \label{#2} }
146     \__head_find_label_aux:w #3\__head_find_label_aux:w
147   }
148 }

```

(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.

```

149 \newcommand\WordSpaceAmount[1]{
150   \glueexpr
151     #1\fontdimen2\the\font
152   plus #1\fontdimen3\the\font
153   minus #1\fontdimen4\the\font
154   \relax
155 }

```

(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.

```
156 \def\frenchspacing{\sfcode`\.\1006\sfcode`\?1005\sfcode`\!1004%
157 \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.

```
158 \def\nonfrenchspacing{\sfcode`\.\3002\sfcode`\?3001\sfcode`\!3000%
159 \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.

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

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

`\AddPunctuation` 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!

```
161 \cs_new_protected:Npn \AddPunctuation #1 {
162   \relax\ifhmode
163     \ifnum\spacefactor>\@m
164       \__head_debug_typeout:n {--->~ punctuation~ '#1'~ not~ added}
165     \else
166       \__head_debug_typeout:n {--->~ punctuation~ '#1'~ added}
167     #1
168     \fi
169   \fi
170 }
```

(End of definition for `\AddPunctuation`. 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 }

```
171 \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

```
172 \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.

```
173 \DeclareTemplateInterface{heading}{display}{9}
174 {
175   , name           : tokenlist
176   , parent-name    : tokenlist
177   , reset-counter  : tokenlist
178   , level          : integer = 0
```

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

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

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

```
180   , placement      : choice {page , top , normal , ragged } = normal
181   , column-spanning : boolean = false
182   , mark-cmd       : function(1) =
183   %
184   % many more keys for layout settings are missing for now
185   , 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.

```

186 , before-vspace : skip = 0pt
187 , penalty       : integer = \c_max_int
188 , after-penalty-vspace : skip = 0pt
189 , 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).

```

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

```

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

```

192 , prefix        : tokenlist = \NoValue

193 , heading-decls  : tokenlist = \normalfont
194 , prefix-decls   : tokenlist =
195 , number-decls   : tokenlist =
196 , title-decls    : tokenlist =
197 , subtitle-decls : tokenlist =
198 , quote-decls    : tokenlist =

199 , headformat-instance : tokenlist = std
200 , number-format      : function(1) = \theheading
201 , contents-extra     : tokenlist =
202 }

```

`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`.

```

203 \DeclareTemplateCopy{heading}{runin}{display}

```

11.7.3 headformat templates interfaces

We have three template: `display`, `hang` and `runin`.

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

```

204 \DeclareTemplateInterface{headformat}{display}{6}
205 {
206 , heading-indent    : length = 0pt
207 , title-format      : function(1) = #1
208 , prefix-number-sep : tokenlist = \WordSpaceAmount{1}
209 , number-title-sep  : tokenlist = 20pt
210 }

```

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

```

211 \DeclareTemplateInterface{headformat}{hang}{6}
212 {
213 , heading-indent    : length = 0pt

```

```

214 , title-format      : function(1) = #1
215 , prefix-number-sep : tokenlist = \WordSpaceAmount{1}
216 , number-title-sep  : tokenlist = 1em
217 }

```

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

```

218 \DeclareTemplateInterface{headformat}{runin}{6}
219 {
220 , heading-indent    : length = 0pt
221 , title-format      : function(1) = #1
222 , prefix-number-sep : tokenlist = \WordSpaceAmount{1}
223 , number-title-sep  : tokenlist = 1em
224 }

```

11.7.4 heading templates code

heading display (*templ.*)

```

225 \DeclareTemplateCode{heading}{display}{9}
226 {
227 , name           = \l__head_name_tl
228 , level          = \l__head_level_int
229 % next two not yet used
230 , parent-name    = \l__head_pname_tl
231 , reset-counter  = \l__head_reset_cnt_tl
232
233 , 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.

```

233 , placement      = {
234   ,page = \__head_debug_typeout:n{ A~ page~ heading }
235         \tl_set:Nn \l__head_placement_tl { page }
236   ,top  = \__head_debug_typeout:n{ A~ top~ heading }
237         \tl_set:Nn \l__head_placement_tl { top }
238   ,normal = \__head_debug_typeout:n{ A~ normal~ heading }
239         \tl_set:Nn \l__head_placement_tl { normal }
240   ,ragged = \__head_debug_typeout:n{ A~ normal~ heading~ (ragged~ style) }
241         \tl_set:Nn \l__head_placement_tl { ragged }
242   }
243
244 , column-spanning = \l__head_column_spanning_bool
245
246 , mark-cmd        = \__head_mark_cmd:n

```

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

```

245 , para-indent    = {
246   ,true  = \@afterindenttrue
247   ,false = \@afterindentfalse
248   }
249 , before-vspace = \l__head_before_skip

```

```

250 , penalty          = \l__head_penalty_int
251 , after-penalty-vspace = \l__head_after_penalty_skip
252 , after-vspace      = \l__head_after_skip
253 , start-code        = \l__head_start_code_tl
254 , final-code        = \l__head_final_code_tl
255 , prefix            = \l__head_typeset_prefix_tl
256 , headformat-instance = \l__head_headformat_instance_tl
257 , heading-decls     = \l__head_heading_decls_tl
258 , prefix-decls      = \l__head_prefix_decls_tl
259 , number-decls      = \l__head_number_decls_tl
260 , title-decls       = \l__head_title_decls_tl
261 , subtitle-decls    = \l__head_subtitle_decls_tl
262 , quote-decls       = \l__head_quote_decls_tl
263 , number-format     = \__head_number_format:n
264 , contents-extra    = \l__head_contents_extra_tl
265 }
266 {
267   \tl_set_eq:Nc \theheading { the \l__head_name_tl }

```

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

```

268   \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}
269   \__head_show_arguments:nnnnnnnnn
270     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

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

```

271   \tl_if_empty:oF {#1} { \SetTemplateKeys{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.

```

272   \bool_if:NT \l__head_column_spanning_bool
273     {
274       \if@twocolumn
275         \str_if_eq:VnF \l__head_placement_tl {top}
276         {
277           \__head_debug_typeout:n {column-spanning~ requires~
278             placement=top.~ Adjusted!}
279           \tl_set:Nn \l__head_placement_tl {top}
280         }
281       \else
282         \bool_set_false:N \l__head_column_spanning_bool
283       \fi
284     }

```

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.

```

285   \tl_if_empty:oT \l__head_start_code_tl
286     { \str_case:VnF \l__head_placement_tl

```

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

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

```

287 {
288   { page }
289   { \tl_set:Nn \l__head_start_code_tl
290     {
291       \if@openright
292         \cleardoublepage
293       \else
294         \clearpage
295       \fi
296       \thispagestyle{plain}%
297       \if@twocolumn
298         \onecolumn
299         \@tempwattrue
300       \else
301         \@tempwafalse
302       \fi
303       \null\vfil
304     }
305   }
306   { top }
307   { \tl_set:Nn \l__head_start_code_tl
308     {
309       \if@openright\cleardoublepage\else\clearpage\fi
310       \thispagestyle{plain}%
311       \global\@topnum\z@
312     }
313   }

```

Straight from the placement definition of `\part`.

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`.

```

314   { ragged }
315   { \cs_set_eq:NN \sec_add_penalty_with_possible_fil:n
316     \sec_add_penalty_with_fil:n
317   }
318 }
319 { \tl_clear:N \l__head_start_code_tl }
320 }
321 %
322 \tl_if_empty:oT \l__head_final_code_tl
323 { \str_case:VnF \l__head_placement_tl
324   {
325     { page }
326     { \tl_set:Nn \l__head_final_code_tl
327       {
328         \vfil\newpage
329         \if@twoside
330           \if@openright
331             \null
332             \thispagestyle{empty}%
333             \newpage

```

```

334         \fi
335     \fi
336     \if@tempwa
337         \twocolumn
338     \fi
339 }
340 }
341 }
342 { \tl_set:Nn \l__head_final_code_tl { \@afterheading } }
343 }

```

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

```

344 \__head_determine_penalty:
345 \__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.

```

346 \__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.

```

347 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
348 \UseTaggingSocket{sec/begin}
349 {{\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}

```

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

```

350 \__head_debug_typeout:n{use~ 'headformat'~instance:~
351                     \l__head_headformat_instance_tl }
352 \use:e {

```

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.

```

353 \bool_if:NT \l__head_column_spanning_bool
354 { \exp_not:n {
355     \@topnewpage [
356         \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
357         { \vspace* \l__head_after_penalty_skip }
358     }
359 }
360 \UseInstance{headformat} { \l__head_headformat_instance_tl }
361 { \exp_not:o \UnusedTemplateKeys }
362 { \exp_not:o { \l__head_typeset_prefix_tl } }
363 { \exp_not:o { \l__head_typeset_number_tl } }
364 { \exp_not:n { #3 } }
365 { \exp_not:o { \use_i:nn #9 } }
366 { \exp_not:o { \use_ii:nn #9 } }

367 \bool_if:NT \l__head_column_spanning_bool
368 {
369     \skip_vertical:N \l__head_after_skip

```

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

```

370         ]
371     }
372 }
373 %
374 % --- handle marks, toc-entry, bookmark, nameref, and label
375 %
376 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
377 %
378 % --- post-heading handling (vertical)

```

Restore `secnumdepth`

```

379 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
380 \par

```

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

```

381 \bool_if:NF \l__head_column_spanning_bool
382 {
383     \nobreak
384     \skip_vertical:N \l__head_after_skip
385 }
386 % --- prepare next paragraph (defaults to \cs{@afterheading}
387 %
388 \l__head_final_code_tl
389 %
390 \ignorespaces
391 }

```

`\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.

```

392 \cs_new_protected:Npn \sec_add_penalty_with_fil:n #1 {
393     \__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`.

```

394 \addpenaltywithfil { #1 }

```

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

```

395 \cs_set_eq:NN
396     \sec_add_penalty_with_possible_fil:n
397     \addpenalty
398 }

```

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.

```

399 \cs_set_eq:NN
400     \sec_add_penalty_with_possible_fil:n
401     \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.

```

402 \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).

```

403   \mode_if_horizontal:T
404   { \mode_if_inner:TF { \@LRmoderr }{ \par } }
405   \legacy_if:nF { @minipage }
406   { \legacy_if:nF { @nobreak }
407     {

```

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

```

408       \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.

```

409       \skip_set_eq:NN \l_tmpa_skip \tex_lastskip:D
410       \dim_compare:nNnTF \l_tmpa_skip = \c_zero_dim
411       { #1 }

```

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

```

412       {
413         \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.

```

414       \dim_set:Nn \l_tmpa_dim
415       {
416         \dim_compare:nNnTF \prevdepth > \maxdepth
417         \maxdepth
418         { \dim_compare:nNnTF \prevdepth = { -1000pt }
419           \c_zero_dim
420           \prevdepth
421         }
422       }
423       \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.

```

424       \vskip -\l_tmpb_skip

```

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

```

425       #1

```

After the penalty (and after a possible `\vfilneg`) we have to readdd 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.

```
426         \dim_compare:nNnF \l_tmpa_dim = \c_zero_dim
427         { \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.

```
428         \vskip \l_tmpa_skip
429     }
430     \group_end:
431 }
432 }
433 }
```

Reimplement `\addpenalty` using the above macro:

```
434 \cs_set_protected:Npn \addpenalty #1 {
435     \__kernel_add_penalty:n { \penalty #1 \scan_stop: }
436 }
```

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

```
437 \cs_new_protected:Npn \addpenaltywithfil #1 {
438     \__kernel_add_penalty:n { \vfil \penalty #1 \vfilneg }
439 }
```

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

`\if@openright`

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

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

```
443     \expandafter\newif\csname if@openright\endcsname
444     \fi
445 }
```

(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.

```
446 \DeclareTemplateCode{heading}{runin}{9}
447 {
448     , name           = \l__head_name_tl
449     , level          = \l__head_level_int
450     % next two not yet used
451     , parent-name    = \l__head_pname_tl
452     , reset-counter  = \l__head_reset_cnt_tl
453     , 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.

```

454 , placement      = {
455     page      = \typeout{ ^^JA~ runin~ page~ placement~ heading-makes-no-sense
456                 (top-used)}
457     \tl_set:Nn \l__head_placement_tl { top }
458 ,top          = \__head_debug_typeout:n{ A~ top~ heading }
459     \tl_set:Nn \l__head_placement_tl { top }
460 ,normal       = \__head_debug_typeout:n{ A~ normal~ heading }
461     \tl_set:Nn \l__head_placement_tl { normal }
462 ,ragged       = \__head_debug_typeout:n{ A~ normal~ heading~ (ragged~ style) }
463     \tl_set:Nn \l__head_placement_tl { ragged }
464     }

465 , column-spanning = \l__head_column_spanning_bool

466 , 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.

```

467 , para-indent    = {
468     ,true  = %\typeout{para-indent~ setting~ ignored}
469     ,false = %\typeout{para-indent~ setting~ ignored}
470     }
471 , before-vspace  = \l__head_before_skip
472 , penalty        = \l__head_penalty_int

```

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

Next key makes no sense either in a runin heading and is not used.

```

473 , after-penalty-vspace = \l__head_after_penalty_skip
474 , after-vspace        = \l__head_after_skip
475 , start-code          = \l__head_start_code_tl
476 , final-code          = \l__head_final_code_tl

477 , prefix              = \l__head_typeset_prefix_tl

478 , headformat-instance = \l__head_headformat_instance_tl

479 , heading-decls       = \l__head_heading_decls_tl
480 , prefix-decls        = \l__head_prefix_decls_tl
481 , number-decls        = \l__head_number_decls_tl
482 , title-decls         = \l__head_title_decls_tl
483 , subtitle-decls      = \l__head_subtitle_decls_tl
484 , quote-decls         = \l__head_quote_decls_tl

485 , number-format       = \__head_number_format:n
486 , contents-extra      = \l__head_contents_extra_tl
487 }
488 {
489   \__head_show_arguments:nnnnnnnnn
490   {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

Revise and separate the two templates better

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

```

491   \tl_set:N\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}

```

define \theheading

```

492 \tl_set_eq:Nc \theheading { the \l__head_name_tl }
493 \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{runin}{#1} }

```

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

```

494 \bool_if:NT \l__head_column_spanning_bool
495 {
496   \__head_debug_typeout:n {column-spanning~ is~
497     not~ (yet)~ supported~ for~ runin~ headings!}
498   \bool_set_false:N \l__head_column_spanning_bool
499 }

500 \tl_if_empty:oT \l__head_start_code_tl
501 {
502   \str_case:Vn \l__head_placement_tl
503   {
504     { top } { \tl_set:Nn \l__head_start_code_tl {\clearpage} } }

505     { ragged }
506     { \cs_set_eq:NN \sec_add_penalty_with_possible_fil:n
507       \sec_add_penalty_with_fil:n
508     }
509   }

```

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

```

510 % { \tl_clear:N \l__head_start_code_tl }
511 }
512 %

```

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 ...

```

513 % \tl_if_empty:oT \l__head_final_code_tl
514 % { \str_case:VnF \l__head_placement_tl
515 % {
516 % { top } { \tl_clear:N \l__head_final_code_tl }
517 % }
518 % { \tl_clear:N \l__head_final_code_tl }
519 % }

520 \__head_determine_penalty:
521 \__head_determine_number_typesetting:N #2
522 \__head_vertical_before_spacing:

```

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

```

523 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
524 \UseTaggingSocket{sec/begin}
525   {{\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
526 \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
527 \def \@svsechd {
528   \__head_debug_typeout:n{use~ 'headformat'-instance:~
529     \l__head_headformat_instance_tl }

```

```

530 \use:e {
531   \UseInstance{headformat} { \l__head_headformat_instance_tl }
532   { \exp_not:o \UnusedTemplateKeys }
533   { \exp_not:o { \l__head_typeset_prefix_tl } }
534   { \exp_not:o { \l__head_typeset_number_tl } }
535   { \exp_not:n { #3 } }
536   { \exp_not:o { \use_i:nn #9 } }
537   { \exp_not:o { \use_ii:nn #9 } }
538 }
539 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}

```

Restore `secnumdepth`

```

540 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
541 }
542 %
543 \@nobreakfalse
544 \global\@noskipsectrue
545 \everypar{%
546   \if@noskipsec
547     \global\@noskipsecfalse
548     {\setbox\z@\lastbox}
549     \clubpenalty\@M
550 %FMi group is in headformat
551 %   \begingroup
552 %   \@svsechd
553 %   \endgroup
554 \unskip

```

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

```

555 \UseTaggingSocket{sec/title/split}
556 \skip_horizontal:N \l__head_after_skip
557 \else
558   \clubpenalty \@clubpenalty
559   \everypar{}%
560 \fi
561 }

```

— prepare next paragraph (does nothing by default)

```

562 \l__head_final_code_tl
563 %
564 \ignorespaces
565 }

```

11.7.5 headformat templates code

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

```

566 \DeclareTemplateCode{headformat}{display}{6} % args: keys,prefix,number,title,subtitle,quot
567 {
568   , heading-indent      = \l__head_heading_indent_dim
569   , title-format        = \__head_title_format:n
570   , prefix-number-sep   = \l__head_prefix_number_sep_tl
571   , number-title-sep    = \l__head_number_title_sep_tl
572 }
573 {

```

```
574 \__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).

```
575 \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{display}{#1} }
```

```
576 \group_begin:
```

```
577 \UseTaggingSocket{sec/title/begin}{\int_use:N\l__head_level_int}{#4}}
```

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

```
578 \normalfont
```

```
579 \interlinepenalty \@M
```

```
580 \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!

```
581 \bool_if:NTF \l__head_unnumbered_bool
```

```
582 {
```

```
583 \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
```

```
584 {
```

```
585 \skip_horizontal:N \l__head_heading_indent_dim
```

```
586 \MakeLinkTarget[\l__head_name_tl]{}
```

```
587 }
```

```
588 {
```

```
589 \MakeLinkTarget[\l__head_name_tl]{\skip_horizontal:N \l__head_heading_indent_dim
```

```
590 }
```

```
591 }
```

```
592 {
```

```
593 \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
```

```
594 {
```

```
595 \skip_horizontal:N \l__head_heading_indent_dim
```

```
596 \MakeLinkTarget{\l__head_name_tl}
```

```
597 }
```

```
598 {
```

```
599 \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent_dim
```

```
600 }
```

```
601 \IfNoValueF{#2}
```

```
602 {
```

```
603 { \l__head_prefix_decls_tl #2 }
```

```
604 \nobreak
```

```
605 \skip_horizontal:n { \l__head_prefix_number_sep_tl }
```

```
606 }
```

```
607 \l__head_number_decls_tl
```

```
608 #3
```

```
609 \par\nobreak
```

```
610 \skip_vertical:n { \l__head_number_title_sep_tl }
```

```
611 }
```

```
612 \l__head_title_decls_tl
```

```
613 \__head_title_format:n {#4}
```

```
614 \par
```

```
615 \UseTaggingSocket{sec/title/end}
```

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

```

616 \group_end:
617 }

```

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

```

618 \DeclareTemplateCode{headformat}{hang}{6} % args: keys,prefix,number,title,subtitle,quotation
619 {
620   , heading-indent      = \l__head_heading_indent_dim
621   , title-format        = \l__head_title_format:n
622   , prefix-number-sep = \l__head_prefix_number_sep_tl
623   , number-title-sep   = \l__head_number_title_sep_tl
624 }
625 {
626   \l__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).

```

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

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

630   \normalfont
631   \interlinepenalty \@M
632   \l__head_heading_decls_tl{}

```

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

```

633   \bool_if:NTF \l__head_unnumbered_bool
634   {
635     \tl_set:Nn\l__head_tmpa_tl
636     {
637       \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
638       {
639         \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget[\l__head_name
640       }
641       {
642         \MakeLinkTarget[\l__head_name_tl]{}
643         \skip_horizontal:N \l__head_heading_indent_dim
644       }
645     }
646   }
647   {
648     \tl_set:Nn \l__head_tmpa_tl
649     {
650       \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
651       {
652         \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget{\l__head_nam
653       }
654       {
655         \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent
656       }
657     }
658     \IfNoValueF{#2}
659     {
660       { \l__head_prefix_decls_tl #2 }

```

```

660         \nobreak
661         \skip_horizontal:n { \l__head_prefix_number_sep_tl }
662     }
663     { \l__head_number_decls_tl #3 }
664     \skip_horizontal:n { \l__head_number_title_sep_tl }
665 }
666 }
667 \UseTaggingSocket{sec/title/hang}
668 {{\int_use:N\l__head_level_int}\l__head_unnumbered_bool{\l__head_tmpa_tl}{#4}}
669 {
670     \@hangfrom
671     {
672         \l__head_tmpa_tl
673     }
674 }
675 \l__head_title_decls_tl
676 \__head_title_format:n {#4}
677 \par
678 \group_end:
679 }

```

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

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

```

680 \DeclareTemplateCode{headformat}{runin}{6} % args: keys,prefix,number,title,subtitle,quotation
681 {
682     , heading-indent      = \l__head_heading_indent_dim
683     , title-format       = \__head_title_format:n
684     , prefix-number-sep  = \l__head_prefix_number_sep_tl
685     , number-title-sep   = \l__head_number_title_sep_tl
686 }
687 {
688     \__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).

```

689 \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }
690 \group_begin:

```

```

691     \normalfont
692     \interlinepenalty \OM
693     \l__head_heading_decls_tl{}

```

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

```

694     \bool_if:NTF \l__head_unnumbered_bool
695     {
696         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
697         {
698             \skip_horizontal:N \l__head_heading_indent_dim
699             \MakeLinkTarget[\l__head_name_tl]{}

```

Setting `\interlinepenalty` makes little sense I think (but that's the way it was in $\text{\LaTeX} 2_{\epsilon}$)

```

700     }
701     {
702         \MakeLinkTarget[\l__head_name_tl]{ }\skip_horizontal:N \l__head_heading_indent_dim
703     }
704 }
705 {
706     \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
707     {
708         \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget{\l__head_name_tl}
709     }
710     {
711         \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent_dim
712     }
713     {
714         \UseTaggingSocket{sec/title/number}{\int_use:N\l__head_level_int}
715         {
716             \IfNoValueF{#2}
717             {
718                 { \l__head_prefix_decls_tl #2 }
719                 \nobreak
720                 \skip_horizontal:n { \l__head_prefix_number_sep_tl }
721             }
722             \l__head_number_decls_tl
723             #3
724         }
725     }
726     \skip_horizontal:n { \l__head_number_title_sep_tl }
727 }
728 \l__head_title_decls_tl
729 \__head_title_format:n {#4}
730 \group_end:
731 }

```

11.7.6 Internal commands used by the template code

`\__head_determine_penalty:`

```

732 \cs_new:Npn \__head_determine_penalty: {

```

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

```

733   \int_compare:nNnT \l__head_penalty_int = \c_max_int
734   { \int_set:Nn \l__head_penalty_int \@secpenalty }
735 }

```

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

`\__head_determine_number_typesetting:N`

```

736 \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.

```

737   \bool_if:NF \l__head_unnumbered_bool

```

```

738     {
739       \bool_set:Nn \l__head_unnumbered_bool
740       { \bool_lazy_or_p:nn
741         { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
742         { \bool_if_p:N #1 }
743       }
744     }

```

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)

```

745     \bool_if:NTF \l__head_unnumbered_bool
746     {
747       \int_gset:Nn \c@secnumdepth{-99}

```

and we set `\l__head_typeset_number_tl` to do nothing.

```

748       \tl_clear:N \l__head_typeset_number_tl
749     }

```

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

```

750     {
751       \@kernel@refstepcounter{ \l__head_name_tl }
752       \protected@edef \l__head_typeset_number_tl
753       {
754         \__head_number_format:n { \l__head_name_tl }
755       }
756     }
757 }

```

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

`\__head_vertical_before_spacing:`

```

758 \cs_new:Npn \__head_vertical_before_spacing: {
759   \tl_if_blank:VTF \l__head_start_code_tl
760   {
761     \if@noskipsec \leavevmode \fi
762     \par
763     \if@nobreak
764       \everypar{}
765     \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.

```

766       \sec_add_penalty_with_possible_fil:n
767       \l__head_penalty_int
768       \addvspace \l__head_before_skip

```

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

```

769       \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
770       { \vspace* \l__head_after_penalty_skip }
771     \fi

```

```

772     }
773     {
774         \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`.

```

775         \bool_if:NF \l__head_column_spanning_bool
776         {
777             \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
778             { \vspace*      \l__head_after_penalty_skip }
779         }
780     }
781 }

```

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

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

```

782 \newcommand\addcontentslinebookmark[3]{
783 \newcommand\addcontentslinebookmarkOff{
784 \newcommand\addcontentslinebookmarkReset{
785 \providecommand\texorpdfstring[2]{#1}

```

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

```

786 \AddToHook{package/hyperref/after}
787 {
788     \RenewCommandCopy\addcontentslinebookmark\Hy@addcontentsline@addbookmark
789     \renewcommand\addcontentslinebookmarkOff
790     {
791         \ifHy@bookmarks
792         \let\addcontentslinebookmarkReset\Hy@bookmarkstrue
793         \else
794         \let\addcontentslinebookmarkReset\relax
795         \fi
796         \Hy@bookmarksfalse
797     }
798 }

```

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

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

```

799 \cs_new:Npn \__head_handle_marks_etc:nnnnn #1#2#3#4#5 {
800     %
801     \tl_set:Nn\@currentlabelname{#4}
802     \IfBlankF {#2}
803     { \__head_mark_cmd:n { #2 } }
804     \IfBlankTF {#1}

```

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

```

805     {
806       \IfBlankF{#3}
807       {
808         \addcontentslinebookmark{toc}{\l__head_name_tl}
809         {
810           \bool_if:NF \l__head_unnumbered_bool
811           {
812             \protect\numberline{ \use:c{ the \l__head_name_tl } }
813           }
814           #3
815         }
816       }
817     }
818     {

```

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.

```

819       \IfBlankT{#3}{\addcontentslinebookmarkOff}
820       \addcontentsline{toc}{ \l__head_name_tl }
821       {
822         \bool_if:NF \l__head_unnumbered_bool
823         {
824           \protect\numberline{ \use:c{ the \l__head_name_tl } }
825         }

```

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

```

826         \texorpdfstring{#1}{#3}
827       }
828       \addcontentslinebookmarkReset
829     }

```

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

```

830   \l__head_contents_extra_tl

```

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

```

831   \__head_debug_typeout:n{--->~label(s):~ \exp_not:n{#5}}
832   #5
833 }

```

(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} }

```

perhaps this should have a hook for packages

then this gets overwritten by every class right now.

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

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

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

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

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

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

The current implementation

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

11.8.1 Instances (sample/default definitions)

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

```

834 \DeclareInstance{heading}{part}{display}
835 {
836   , name           = part
837   , level          = -1
838   , placement      = page
839   , after-penalty-vspace = 0pt
840   , prefix         = \partname
841   , number-format  = \thepart
842   , heading-decls  = \centering\bfseries\huge
843   , title-decls    = \Huge
844   , headformat-instance = part

```

```

845 , mark-cmd      = \partmark {#1}
846 }

```

heading chapter (*inst.*)

```

847 \DeclareInstance{heading}{chapter}{display}
848 {
849   , name          = chapter
850   , level         = 0
851   , placement     = top
852   , column-spanning = true
853   , after-penalty-vspace = 50pt
854   , after-vspace  = 40pt
855   , prefix        = \@chapapp
856   , number-format = \thechapter
857   , heading-decls = \raggedright \parindent0pt \bfseries \huge
858   , title-decls   = \Huge
859   , headformat-instance = chapter
860   , mark-cmd      = \chaptermark {#1}
861   , contents-extra = \addtocontents{lof}{\addvspace{10pt}}
862                   \addtocontents{lot}{\addvspace{10pt}}
863 }

```

`\@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.

```

864 \AtBeginDocument{
865   \cs_if_exist:NF \@chapapp { \cs_new:Npn \@chapapp {Chapter} } }

```

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

heading section (*inst.*)

```

866 \DeclareInstance{heading}{section}{display}
867 {
868   , name          = section
869   , level         = 1
870   , mark-cmd      = \sectionmark {#1}
871   , before-vspace = 3.5ex plus 1ex minus .2ex
872   , after-vspace  = 2.3ex plus .2ex
873   , heading-decls = \normalfont\Large\bfseries
874   , headformat-instance = section
875 }

```

heading subsection (*inst.*)

```

876 \DeclareInstance{heading}{subsection}{display}
877 {
878   , name          = subsection
879   , parent-name   = section
880   , level         = 2
881   , mark-cmd      = \subsectionmark {#1}
882   , before-vspace = 3.25ex plus 1ex minus .2ex
883   , after-vspace  = 1.5ex plus .2ex
884   , heading-decls = \normalfont\large\bfseries
885   , headformat-instance = subsection
886 }

```

heading subsubsection (*inst.*)

```

887 \DeclareInstance{heading}{subsubsection}{display}
888 {
889   , name           = subsubsection
890   , parent-name    = subsection
891   , level          = 3
892   , before-vspace  = 3.25ex plus 1ex minus .2ex
893   , after-vspace   = 1.5ex plus .2ex
894   , heading-decls  = \normalfont\normalsize\bfseries
895   , headformat-instance = subsubsection
896 }

```

heading paragraph (*inst.*)

```

897 \DeclareInstance{heading}{paragraph}{runin}
898 {
899   , name           = paragraph
900   , parent-name    = subsubsection
901   , level          = 4
902   , before-vspace  = 3.25ex \@plus1ex \@minus.2ex
903   , after-vspace   = 1em
904   , heading-decls  = \normalfont\normalsize\bfseries
905   , mark-cmd       = \paragraphmark{#1}
906   , headformat-instance = paragraph
907 }
908

```

heading subparagraph (*inst.*)

```

909 \DeclareInstance{heading}{subparagraph}{runin}
910 {
911   , name           = subparagraph
912   , parent-name    = paragraph
913   , level          = 5
914   , before-vspace  = 3.25ex \@plus1ex \@minus.2ex
915   , after-vspace   = 1em
916   , heading-decls  = \normalfont\normalsize\bfseries
917   , mark-cmd       = \subparagraphmark{#1}
918   , headformat-instance = subparagraph
919 }
920

```

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

```

921 \DeclareInstance{headformat}{part}{display}
922 {
923   , heading-indent    = 0pt
924   , prefix-number-sep = \WordSpaceAmount{1}
925 }
926 \DeclareInstanceCopy{headformat}{chapter}{part}

```

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

headformat section (*inst.*)

headformat subsection (*inst.*)

headformat subsubsection (*inst.*)

```

927 \DeclareInstance{headformat}{std}{hang}

```

```

928 {
929   , heading-indent = Opt
930 }

931 \DeclareInstanceCopy{headformat}{section}{std}
932 \DeclareInstanceCopy{headformat}{subsection}{std}
933 \DeclareInstanceCopy{headformat}{subsubsection}{std}

```

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

```

headformat subparagraph (inst.)
934 \DeclareInstance{headformat}{paragraph}{runin}
935 {
936   , heading-indent = Opt
937 }

938 \DeclareInstance{headformat}{subparagraph}{runin}
939 {
940   , heading-indent = \parindent
941 }

```

`headformat section-special (inst.)`

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

```

942 \DeclareInstance{headformat}{section-special}{hang}
943 {
944   , heading-indent = 4em
945   , title-format   = {#1!}
946 }

```

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`.

```

947 \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.

```

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

```

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

```

951     \@tempskipa #4\relax
952     \@afterindenttrue
953     \ifdim \@tempskipa <\z@
954         \@tempskipa -\@tempskipa
955         \@afterindentfalse
956     \fi

```

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

```

957     \@tempskipb #5\relax
958     \ifdim \@tempskipb >\z@
959         \use:e {
960             \DeclareInstance{heading}{#1-@startsection}{display}{
961                 , name          = #1
962                 , level         = #2
963                 , mark-cmd      = \exp_not:c {#1mark} {##1}
964                 , before-vspace = \the\@tempskipa
965                 , after-vspace  = \the\@tempskipb
966                 , para-indent   = \if@afterindent true \else false\fi
967                 , heading-decls = \exp_not:n {#6}

```

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

```

968         , headformat-instance = #1-@startsection
969     }}

```

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

```

970     \DeclareInstance{headformat}{#1-@startsection}{hang}{heading-indent = #3,title-
format={#6{##1}}}}
971     \else
972         \@tempskipb=-\@tempskipb
973         \use:e {
974             \DeclareInstance{heading}{#1-@startsection}{runin}{
975                 , name          = #1
976                 , level         = #2
977                 , mark-cmd      = \exp_not:c {#1mark} {##1}
978                 , before-vspace = \the\@tempskipa
979                 , after-vspace  = \the\@tempskipb
980                 , heading-decls = \exp_not:n{#6}
981                 , headformat-instance = #1-@startsection
982             }}
983             \DeclareInstance{headformat}{#1-@startsection}{runin}{heading-indent = #3,title-
format={#6{##1}}}}
984         \fi
985     }
986     \ParseLaTeXeHeading {#1-@startsection}{#7}{#8}{#9}
987 }

```

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

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

```

988 \NewCommandCopy\kernel@startsection\@startsection
989 \AddToHook{class/after}[head/@startsection]
990 {\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`

```

991 \RenewDocumentCommand\secdef{mmsO{#5}m}
992 {
993   \str_case:enF {\cs_to_str:N#1}
994   {
995     \@part}
996     {
997       \IfNoValueTF{#4}
998       {\ParseLaTeXeHeading{part}{#3} {start-code=\relax} {#5}}
999       {
1000         \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
1001         \ExpandArgs{nne}\ParseLaTeXeHeading{part}{#3}{start-code=\relax,\exp_not:o{\l__head
1002         }}
1003         \@latex@warning{\noexpand\secdef~detected~in~\noexpand\part~command.\MessageBreak
1004         \noexpand\part~will~be~redefined~to~use~templates.\MessageBreak
1005         This~may~change~the~layout!}
1006         \DeclareDocumentCommand \part {s = {shorttitle}o m}
1007         { \ParseLaTeXeHeading {part} {##1} {##2} {##3} }
1008       }
1009     }@chapter}
1010     {
1011       \IfNoValueTF{#4}
1012       { \ParseLaTeXeHeading{chapter}{#3} {#4} {#5} }
1013       {
1014         \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
1015         \ExpandArgs{nno}\ParseLaTeXeHeading{chapter}{#3}{\l__head_tmpa_tl} {#5}
1016       }
1017       \@latex@warning{\noexpand\secdef~detected~in~\noexpand\chapter~command.\MessageBreak
1018       \noexpand\chapter~will~be~redefined~to~use~templates.\MessageBreak
1019       This~may~change~the~layout!}
1020       \DeclareDocumentCommand \chapter {s = {shorttitle}o m}
1021       { \ParseLaTeXeHeading {chapter} {##1} {##2} {##3} }
1022     }
1023   }
1024   {
1025     \tag_if_active:TF\@latex@error\@latex@warning
1026     {\noexpand\secdef~with~unknown~argument~\noexpand#1~found.\MessageBreak
1027     The~command~can~not~be~adapted~on~the~fly~to~support~tagging.\MessageBreak
1028     The~heading~command~using~this~\noexpand\secdef~should~be~
1029     reimplemented\MessageBreak with~templates}{ }
1030     \IfBooleanTF{#3}{#2{#5}}{#1[#4]{#5}}
1031   }

```

```
1032 }
```

A helper command to reprocess the argument as key-val

```
1033 \NewDocumentCommand\__head_process_shorttitle:nN{={shorttitle}mm}
1034 { \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.

```
1035 \AddToHook{class/article/after}[head/example]
1036 {
1037   \DeclareInstance{heading}{part}{display}
1038   {
1039     , name           = part
1040     , level          = -1
1041     , before-vspace = 4ex
1042     , after-vspace  = 3ex
1043     , mark-cmd       = \partmark {#1}
1044     , prefix         = \partname
1045     , number-format = \thepart
1046     , heading-decls = \raggedright\bfseries\Large
1047     , title-decls   = \huge
1048     , headformat-instance = part
1049   }
1050   \DeclareInstance{headformat}{part}{display}
1051   {
1052     , heading-indent   = 0pt
1053     , number-title-sep = 0pt
1054     , prefix-number-sep = \WordSpaceAmount{1}
1055   }
1056   \DeclareDocumentCommand \part {s = {shorttitle} o m}
1057   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1058   \__head_setup_default_heading:
1059 }

1060 \AddToHook{class/report/after}[head/example]
1061 {
1062   \DeclareDocumentCommand \part {s = {shorttitle} o m}
1063   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1064   \DeclareDocumentCommand \chapter {s = {shorttitle} o m}
1065   {
1066     \ParseLaTeXeHeading {chapter} {#1} {#2} {#3}
1067   }
1068   \__head_setup_default_heading:
1069 }

1070 \AddToHook{class/book/after}[head/example]
1071 {
1072   \DeclareDocumentCommand \part {s = {shorttitle} o m}
1073   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
1074   \DeclareDocumentCommand \chapter {s = {shorttitle} o m}
1075   {
```

```

1076         \if@mainmatter
1077         \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
1078         \else
1079         \IfBooleanTF{#1}
1080         {% starred:
1081         \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
1082         }
1083         {\IfNoValueTF{#2}
1084         {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {shorttitle={#3}} {#3}}
1085         {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}
1086         }
1087         \fi
1088     }
1089     \__head_setup_default_heading:
1090 }

1091 \cs_new_protected:Npn \__head_setup_default_heading:
1092 {

\section

1093     \DeclareDocumentCommand \section {s ={\shorttitle}o m}
1094     { \ParseLaTeXeHeading {section} {##1} {##2} {##3} }

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

\subsection

1095     \DeclareDocumentCommand \subsection {s ={\shorttitle}o m}
1096     { \ParseLaTeXeHeading {subsection} {##1} {##2} {##3} }

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

\subsubsection

1097     \DeclareDocumentCommand \subsubsection {s ={\shorttitle}o m}
1098     { \ParseLaTeXeHeading {subsubsection} {##1} {##2} {##3} }

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

\paragraph

1099     \DeclareDocumentCommand \paragraph {s ={\shorttitle}o m}
1100     { \ParseLaTeXeHeading {paragraph} {##1} {##2} {##3} }

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

\subparagraph

1101     \DeclareDocumentCommand \subparagraph {s ={\shorttitle}o m}
1102     { \ParseLaTeXeHeading {subparagraph} {##1} {##2} {##3} }
1103     } %end of command

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

1104 \</package>

```

12 Issues and problems noticed along the way

12.1 Local `\DeclareInstance`

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

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

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

12.2 `\SetCurrentTemplateKeys`

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

12.3 O-expansion of `\UseInstance` arguments

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

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

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

12.4 Switch `\openright` is not defined by core

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

12.5 Indexheading at least for `l3doc` is wrong now

Needs checking.

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		chaptermark	11, 860
\!	156, 158	\cleardoublepage	292, 309
\,	157, 159	\clearpage	10, 41, 294, 309, 504
\.	156, 158	\clubpenalty	549, 558
\:	157, 159	\cs	386
\;	157, 159	cs commands:	
\?	156, 158	\cs_gset_protected:Npe	23, 25
\@startsection	16	\cs_if_exist:NTF	865
A		\cs_new:Npn	31,
\addcontents	12	40, 125, 140, 732, 736, 758, 799, 865	
\addcontentsline	17, 42, 43, 820	\cs_new_eq:NN	9, 10
\addcontentslinebookmark	782, 808	\cs_new_protected:Npn	11, 16, 21,
\addcontentslinebookmarkOff	782, 819	28, 29, 68, 160, 161, 392, 402, 437, 1091	
\addcontentslinebookmarkOn	782	\cs_set_eq:NN	315, 395, 399, 506
\addcontentslinebookmarkReset	784, 792, 794, 828	\cs_set_protected:Npn	434
\addpenalty	29, 31–33, 41, 397, 401, 402	\cs_to_str:N	993
\addpenaltywithfil	394, 402	\csname	443
\AddPunctuation	161	D	
\addtocontents	861, 862	\DebugHeadingsOff	17, 28
\AddToHook 67, 440, 786, 989, 1035, 1060, 1070		\DebugHeadingsOn	17, 28
\addvspace	32, 768, 861, 862	\DeclareDocumentCommand	947,
\AtBeginDocument	864	1006, 1020, 1056, 1062, 1064, 1072,	
B		1074, 1093, 1095, 1097, 1099, 1101	
\begingroup	551	\DeclareInstance	51, 834, 847, 866,
\bfseries	842, 857, 873, 884, 894, 904, 916, 1046	876, 887, 897, 909, 921, 927, 934,	
bool commands:		938, 942, 960, 970, 974, 983, 1037, 1050	
\bool_gset_false:N	18	\DeclareInstanceCopy	926, 931, 932, 933
\bool_gset_true:N	13	\DeclareTemplateCode	26, 225, 446, 566, 618, 680
\bool_if:NTF	24,	\DeclareTemplateCopy	26, 203
26, 97, 272, 353, 367, 381, 494,		\DeclareTemplateInterface	173, 204, 211, 218
581, 633, 694, 737, 745, 775, 810, 822		\def	156, 158, 527
\bool_if_p:N	742	dim commands:	
\bool_lazy_or_p:nn	740	\dim_compare:nNnTF	356, 410, 416, 418, 426,
\bool_new:N	8, 59	583, 593, 637, 650, 696, 706, 769, 777	
\bool_set:Nn	739	\dim_set:Nn	414
\bool_set_false:N	85, 282, 498	\l_tmpa_dim	414, 423, 426, 427
\bool_set_true:N	79	\c_zero_dim	410, 419, 426
\BooleanFalse	9, 97	\DocumentMetadata	2, 23
\BooleanTrue	9, 97, 1081, 1084, 1085	E	
C		\EditTemplateDefaults	26
\centering	842	\else	165, 281, 293, 300,
\chapter	4, 8, 15, 16,	309, 442, 557, 765, 793, 966, 971, 1078	
43–45, 48, 1017, 1018, 1020, 1064, 1074		\endcsname	441, 443

`\endgroup` 553
`\everypar` 35, 545, 559, 764
exp commands:
 `\exp_not:N` 95, 963, 977
 `\exp_not:n` 33, 34, 35,
 36, 37, 38, 42, 43, 44, 45, 46, 47, 48,
 49, 50, 71, 72, 96, 98, 99, 100, 101,
 102, 103, 104, 105, 138, 354, 361,
 362, 363, 364, 365, 366, 532, 533,
 534, 535, 536, 537, 831, 967, 980, 1001
`\expandafter` 443
`\ExpandArgs` 1001, 1015

F

`\fbox` 4
`\fi` 168, 169, 283,
 295, 302, 309, 334, 335, 338, 444,
 560, 761, 771, 795, 956, 966, 984, 1087
`\filbreak` 31, 41
`\font` 151, 152, 153
`\fontdimen` 151, 152, 153
`\frenchspacing` 23, 24, 156

G

`\global` 311, 544, 547
`\glueexpr` 150
group commands:
 `\group_begin:` 408, 576, 628, 690
 `\group_end:` 430, 616, 678, 730

H

head commands:
 `\head_debug_off:` 17, 11, 16, 29
 `\head_debug_on:` 17, 11, 11, 28
head internal commands:
 `\l_head_after_penalty_skip`
 251, 356, 357, 473, 769, 770, 777, 778
 `\l_head_after_skip`
 252, 369, 384, 474, 556
 `\l_head_before_skip` ... 249, 471, 768
 `\l_head_bookmark_tl`
 53, 78, 84, 101, 111
 `\l_head_column_spanning_bool` ..
 30, 243, 272,
 282, 353, 367, 381, 465, 494, 498, 775
 `\l_head_contents_extra_tl`
 264, 486, 830
 `\_head_debug:n` 9, 9, 23
 `\g_head_debug_bool` .. 8, 13, 18, 24, 26
 `\_head_debug_gset:` 11, 14, 19, 21
 `\_head_debug_typeof:n` 9,
 10, 25, 32, 33, 34, 35, 36, 37, 38,
 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
 69, 70, 93, 128, 133, 138, 144, 164,
 166, 234, 236, 238, 240, 277, 350,
 393, 458, 460, 462, 496, 528, 831, 949
 `\_head_determine_number_-`
 typesetting:N ... 345, 521, 736, 736
 `\_head_determine_penalty:`
 344, 520, 732, 732
 `\l_head_final_code_tl`
 27, 28, 35, 254, 322,
 326, 342, 388, 476, 513, 516, 518, 562
 `\_head_find_label:w` 73, 125, 125
 `\_head_find_label_aux:w`
 22, 136, 140, 146
 `\_head_handle_marks_etc:nnnnn` .
 376, 539, 799, 799
 `\l_head_headformat_instance_tl`
 256, 351, 360, 478, 529, 531
 `\l_head_heading_decls_tl`
 257, 479, 580, 632, 693
 `\l_head_heading_indent_dim`
 568, 583, 585, 589, 593, 595,
 599, 620, 637, 639, 643, 650, 652,
 655, 682, 696, 698, 702, 706, 708, 711
 `\l_head_instance_keys_tl`
 21, 60, 91, 92, 96
 `\l_head_label_tl`
 20, 22, 60, 103, 123, 129, 134, 145
 `\l_head_level_int`
 228, 347, 349, 449,
 523, 525, 526, 577, 629, 668, 714, 741
 `\_head_mark_cmd:n` 244, 466, 803
 `\l_head_name_tl`
 227, 267, 448, 492, 586, 589, 596,
 599, 639, 642, 652, 655, 699, 702,
 708, 711, 751, 754, 808, 812, 820, 824
 `\l_head_nameref_tl` .. 56, 87, 102, 114
 `\l_head_number_decls_tl`
 259, 481, 607, 663, 722
 `\_head_number_format:n` 263, 485, 754
 `\l_head_number_title_sep_tl` ...
 571, 610, 623, 664, 685, 726
 `\l_head_penalty_int`
 250, 472, 733, 734, 767
 `\l_head_placement_tl`
 60, 235, 237, 239, 241, 275, 279,
 286, 323, 457, 459, 461, 463, 502, 514
 `\l_head_pname_tl` 230, 451
 `\l_head_prefix_decls_tl`
 258, 480, 603, 659, 718
 `\l_head_prefix_number_sep_tl` ..
 570, 605, 622, 661, 684, 720
 `\_head_process_shorttitle:nN` ..
 1000, 1014, 1033
 `\l_head_quote_decls_tl` ... 262, 484
 `\l_head_quote_tl` ... 58, 89, 105, 122

L		<code>\ParseLaTeXeHeading</code>
<code>\label</code> <i>3, 9, 20–23, 73, 123, 125, 134, 140, 145</i>		 <i>7, 9, 20, 21, 43, 68, 986, 998,</i>
<code>\Large</code>	873, 1046	1001, 1007, 1012, 1015, 1021, 1057,
<code>\large</code>	884	1063, 1066, 1073, 1077, 1081, 1084,
<code>\lastbox</code>	548	1085, 1094, 1096, 1098, 1100, 1102
<code>\lastskip</code>	32	<code>\part</code> <i>3, 4, 8, 15, 16, 20, 28, 44,</i>
<code>\leavevmode</code>	761	<i>48, 1003, 1004, 1006, 1056, 1062, 1072</i>
legacy commands:		<code>\partmark</code> <i>3</i>
<code>\legacy_if:nTF</code>	405, 406	<code>\partmark</code> <i>67, 845, 1043</i>
<code>\let</code>	792, 794	<code>\partname</code> <i>840, 1044</i>
<code>\ltlabsecIIdate</code>	5	<code>\penalty</code> <i>31, 32, 435, 438</i>
<code>\ltlabsecIIversion</code>	6	<code>\prevdepth</code> <i>32, 416, 418, 420</i>
M		<code>\protect</code> <i>812, 824</i>
<code>\MakeLinkTarget</code> <i>16, 586, 589, 596, 599,</i>		<code>\providecommand</code> <i>67, 785</i>
<i>639, 642, 652, 655, 699, 702, 708, 711</i>		<code>\ProvidesExplPackage</code> <i>3</i>
<code>\MakeUppercase</code>	<i>13, 14, 48</i>	
<code>\markboth</code>	67	Q
<code>\maxdepth</code>	<i>32, 416, 417</i>	quark commands:
<code>\MessageBreak</code>	1003,	<code>\q_no_value</code> <i>22, 73</i>
<i>1004, 1017, 1018, 1026, 1027, 1029</i>		<code>\quark_if_no_value:nTF</code> <i>126, 142</i>
mode commands:		
<code>\mode_if_horizontal:TF</code>	403	R
<code>\mode_if_inner:TF</code>	404	<code>\raggedright</code> <i>857, 1046</i>
N		<code>\refstepcounter</code> <i>16, 41</i>
<code>\nameref</code>	3	<code>\relax</code> . . . <i>154, 162, 794, 951, 957, 998, 1001</i>
<code>\newcommand</code>	<i>149, 782, 783, 784</i>	<code>\renewcommand</code> <i>789</i>
<code>\NewCommandCopy</code>	988	<code>\RenewCommandCopy</code> <i>788, 990</i>
<code>\NewDocumentCommand</code>	<i>9, 1033</i>	<code>\RenewDocumentCommand</code> <i>991</i>
<code>\newif</code>	<i>52, 443</i>	<code>\RestoreLastSkip</code> <i>11</i>
<code>\newpage</code>	<i>328, 333</i>	
<code>\NewTemplateType</code>	<i>171, 172</i>	S
<code>\nobreak</code>	<i>383, 604, 609, 660, 719</i>	<code>\SaveLastSkip</code> <i>11</i>
<code>\noexpand</code> <i>1003, 1004, 1017, 1018, 1026, 1028</i>		scan commands:
<code>\NoNalue</code>	8	<code>\scan_stop:</code> <i>435</i>
<code>\nonfrenchspacing</code>	<i>23, 24, 156</i>	sec commands:
<code>\nopunct</code>	<i>24, 160</i>	<code>\sec_add_penalty_with_fil:n</code>
<code>\normalcolor</code>	36	 <i>31, 41, 316, 392, 392, 507</i>
<code>\normalfont</code>	<i>11, 12, 193,</i>	<code>\sec_add_penalty_with_possible_-</code>
<i>578, 630, 691, 873, 884, 894, 904, 916</i>		<code>fil:n</code> <i>29,</i>
<code>\normalsize</code>	<i>894, 904, 916</i>	<i>31, 41, 315, 392, 396, 400, 506, 766</i>
<code>\NoValue</code>	<i>9–11, 192</i>	<code>\secdef</code> <i>16</i>
<code>\null</code>	<i>303, 331</i>	<code>\secdef</code> <i>4, 8, 16, 44, 45, 48, 991</i>
<code>\numberline</code>	<i>812, 824</i>	<code>\section</code> . . . <i>4, 5, 7, 8, 14, 15, 43, 47, 1093</i>
O		<code>\sectionmark</code> <i>11, 870</i>
<code>\onecolumn</code>	298	<code>\setbox</code> <i>548</i>
<code>\openright</code>	52	<code>\SetCurrentTemplateKeys</code> <i>28, 51</i>
P		<code>\SetTemplateKey</code> <i>28</i>
<code>\par</code>	<i>380, 404, 609, 614, 677, 762</i>	<code>\SetTemplateKeys</code> . . <i>271, 493, 575, 627, 689</i>
<code>\paragraph</code>	<i>14–16, 1099</i>	<code>\sfcode</code> <i>23, 156, 157, 158, 159</i>
<code>\paragraphmark</code>	905	skip commands:
<code>\parindent</code>	<i>857, 940</i>	<code>\skip_add:Nn</code> <i>423</i>
		<code>\skip_horizontal:N</code>
		 <i>556, 585, 589, 595, 599,</i>
		<i>639, 643, 652, 655, 698, 702, 708, 711</i>

<code>\skip_horizontal:n</code>	605, 661, 664, 720, 726
<code>\skip_set_eq:NN</code>	409, 413
<code>\skip_vertical:N</code>	369, 384
<code>\skip_vertical:n</code>	610
<code>\l_tmpa_skip</code>	409, 410, 413, 428
<code>\l_tmpb_skip</code>	32, 413, 423, 424
<code>\c_zero_skip</code>	583, 593, 637, 650, 696, 706
<code>\spacefactor</code>	23, 24, 160, 163
<code>\specialsection</code>	47
str commands:	
<code>\str_case:nn</code>	502
<code>\str_case:nnTF</code>	286, 323, 514, 993
<code>\str_if_eq:nnTF</code>	275
<code>\string</code>	950
<code>\subparagraph</code>	14–16, 1101
<code>\subparagraphmark</code>	917
<code>\subsection</code>	14, 15, 1095
<code>\subsectionmark</code>	881
<code>\subsubsection</code>	14–16, 1097
T	
tag commands:	
<code>\tag_if_active:TF</code>	1025
template types:	
<code>headformat</code>	172
<code>heading</code>	171
templates:	
<code>headformat display</code>	204, 566
<code>headformat hang</code>	211, 618
<code>headformat runin</code>	218, 680
<code>heading display</code>	173, 225
<code>heading runin</code>	203, 446
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@</code>	24
<code>\@LRmoderr</code>	404
<code>\@M</code>	549, 579, 631, 692
<code>\@addpunct</code>	24
<code>\@afterheading</code>	342
<code>\@afterindentfalse</code>	247, 955
<code>\@afterindenttrue</code>	246, 952
<code>\@chapapp</code>	45, 855, 864
<code>\@chappap</code>	25
<code>\@chapter</code>	4
<code>\@clubpenalty</code>	558
<code>\@currentlabelname</code>	801
<code>\@firstoftwo</code>	9
<code>\@hangfrom</code>	670
<code>\@kernel@refstepcounter</code>	751
<code>\@latex@error</code>	1025
<code>\@latex@warning</code>	1003, 1017, 1025
<code>\@m</code>	163
<code>\@minus</code>	902, 914
<code>\@nobreakfalse</code>	543
<code>\@noskipsecfalse</code>	547
<code>\@noskipsectrue</code>	544
<code>\@part</code>	4
<code>\@plus</code>	902, 914
<code>\@secntformat</code>	10
<code>\@secondoftwo</code>	9
<code>\@secpenalty</code>	11, 12, 40, 734
<code>\@startsection</code>	4, 5, 7, 8, 10, 16, 43, 44, 47, 48, 51, 947, 988, 990
<code>\@svsechd</code>	527, 552
<code>\@tempskipa</code>	951, 953, 954, 964, 978
<code>\@tempskipb</code>	957, 958, 965, 972, 979
<code>\@tempswa</code>	28
<code>\@tempswafalse</code>	301
<code>\@tempswatrue</code>	299
<code>\@topnewpage</code>	28, 30, 41, 355
<code>\@topnum</code>	311
<code>\c@secnumdepth</code>	40, 268, 379, 491, 540, 741, 747
<code>\Hy@addcontentsline@addbookmark</code>	788
<code>\Hy@bookmarksfalse</code>	796
<code>\Hy@bookmarkstrue</code>	792
<code>\if@afterindent</code>	966
<code>\if@mainmatter</code>	1076
<code>\if@nobreak</code>	763
<code>\if@noskipsec</code>	546, 761
<code>\if@openright</code>	291, 309, 330, 440
<code>\if@tempswa</code>	336
<code>\if@twocolumn</code>	274, 297
<code>\if@twoside</code>	329
<code>\ifHy@bookmarks</code>	791
<code>\kernel@startsection</code>	988, 990
<code>\protected@edef</code>	752
<code>\z@</code>	311, 548, 953, 958
tex commands:	
<code>\tex_lastskip:D</code>	409
<code>\texorpdfstring</code>	17, 43, 785, 826
<code>\the</code>	151, 152, 153, 964, 965, 978, 979
<code>\the⟨name⟩</code>	11
<code>\thechapter</code>	19, 856
<code>\theheading</code>	3
<code>\theheading</code> ..	11–13, 34, 66, 200, 267, 492
<code>\thepart</code>	841, 1045
<code>\thesection</code>	12, 13, 19
<code>\thispagestyle</code>	296, 310, 332
<code>\titleformat</code>	16
<code>\titleformat</code>	16
<code>\titlespacing</code>	16
<code>\titlespacing</code>	16
tl commands:	
<code>\c_novalue_tl</code>	21, 88, 89
<code>\tl_clear:N</code>	76, 77, 78, 91, 129, 319, 510, 516, 518, 748
<code>\tl_if_blank:nTF</code>	759

