The nccsect package[*][†]

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2006/01/19

Contents

[*]This file has version number v1.5, last revised 2006/01/19.

[†]Great thanks to Denis G. Samsonenko `<d.g.samsonenko@gmail.com>` who proposed many significant improvements to the package.

1 The Scope and Objectives

The package provides a new implementation of sections, captions, and toc-entries independent on the LaTeX kernel. The reasons for this are concerned with the following disadvantages of the standard LaTeX implementation:

1 Standard LaTeX sectioning commands can prepare display sections in the single style: justified paragraph with hang indented number. To change this style to another one (centered, par-indented, or else), you need to re-implement the internal `\@sect` command. It is no control for this style on user's level.

2 If you want to customize the presentation a number in a section (for example, put a paragraph mark § before a number or put a point after a number), you at least need to re-implement the `\@sect` command.

3 The sectioning commands provide no straightforward control for running headings. The marking commands like the `\sectionmark` solve this problem partially. Using them within parameter of sectioning command, you can change the mark properly, but this solution does not work in complicated documents which use first and last marks appearing on a page. The safe solution consists in direct replacement a mark prepared within the `\@sect` command to a custom mark.

4 Special efforts are required to pass a section without number to the header and to the toc-list. There is no simple solution providing this action.

5 Captions for tables and figures are prepared in just the same way, although, they are usually used in different places of floating environments: table captions start *before* a table, but figure captions go *after* a figure. So, the vertical skip inserted before a caption is unnecessary for table captions. The right solution is to design captions for different float types in different ways.

6 The star-form of captions is absent. It is useful when a document contains an alone figure or table. Moreover, in fiction books, unnumbered captions useful.

7 The design of toc-entries is hard for modifications. It is much better to calculate the skips in toc-entries on the base of prototyping technique instead of hard-coding them with absolute values. Moreover, the skips for nested sections must depend on higher level skips. For example, if we change skips for a section entry, the skips for subsection entries should be adjusted automatically.

The package eliminates above-mention disadvantages of the standard LaTeX implementation. The commands implemented in it are divided into two levels: user level and design level. The user-level commands are intended for use within a document and the design-level commands are directed to class and package writers.

2 User Interface

The table below shows sectioning commands provided with standard LaTeX classes. Every section has a *level* (an integer number). Sections can be printed in one of two modes: *display* or *running* mode. Display section is prepared as a separate justified paragraph having a hang indent if a section has a number. Running section starts a paragraph.

Command	Level	Mode
`\part`	−1 or 0[1]	display
`\chapter`	0[2]	display
`\section`	1	display
`\subsection`	2	display
`\subsubsection`	3	display
`\paragraph`	4	running
`\subparagraph`	5	running

`\startsection` The package redefines all standard sectioning commands. Along with the commands shown in the table above, you can use the following uniform notations:

`\startsection{`⟨*level*⟩`}[`⟨*toc-entry*⟩`]{`⟨*title*⟩`}` or
`\startsection{`⟨*level*⟩`}*{`⟨*title*⟩`}`

The ⟨*level*⟩ is a level of section. A negative level produces a part. The first command produces a numbered section (if the numbering depth allows this) and the last one produces a section without number. As for the standard LaTeX sectioning, the first variant of the `\startsection` command additionally passes their arguments to the section mark command (if the mark command exists) and to the aux-file. The last variant does no additional actions.

NOTE: The package allows declaring additional section levels. They, of course, have no predefined alias names as standard section levels.

`\sectionstyle` The `\sectionstyle[`⟨*type*⟩`]{`⟨*style*⟩`}` command allows change a style of subsequent display sections of the given ⟨*type*⟩:

`main`	the section of zero level (`\part` or `\chapter`);
`section`	the `\section`;
`subsection`	the `\subsection`;
`subsubsection`	the `\subsubsection`;
`paragraph`	the `\paragraph`;
`subparagraph`	the `\subparagraph`;
`section@vi`	the section of 6th level, and so on.

[1]The `\part` command has zero level in article-like classes and has the negative level in book-like classes. In book-like classes a part is prepared on a separate page.
[2]The `\chapter` command is defined in book-like classes only.

If the ⟨*type*⟩ parameter is omitted, the command acts on all subsequent display sections expect those having a specialized style. The following styles are predefined:

`hangindent`	standard LaTeX style (default);
`hangindent*`	the same as `hangindent`, but ragged right;
`parindent`	title indented on `\parindent`;
`parindent*`	the same as `parindent`, but ragged right;
`hangparindent`	`\parindent` indented with hang number;
`hangparindent*`	the same as `hangparindent`, but ragged right;
`center`	centered title;
`centerlast`	justified title without indent whose last line is centered.

You can apply the `\sectionstyle` so many times in the document as you want. This command complies with standard LaTeX scoping rules.

NOTE: The section style acts on display sections that were prepared with the dynamic alignment (see Section 4). By default, the sections of levels from 0 to 3 have the dynamic alignment. The section of zero level has no hang indentation.

`\sectiontagsuffix` The `\sectiontagsuffix[`⟨*type*⟩`]{`⟨*style*⟩`}` command allows change a suffix inserted after number tag for sections of the given ⟨*type*⟩. If the ⟨*type*⟩ parameter is omitted, the command acts on all subsequent sections expect those having a specialized tag suffix.

`\indentaftersection` `\noindentaftersection` The paragraph indentation after a display section is controlled with the `\indentaftersection` and `\noindentaftersection` commands. The first one allows and the last one suppresses indentation after section. The commands act on the subsequent display sections in the scope of their use.

`\aftersectionvspace` If a document contains two subsequent sectioning commands (for example, `\section` and `\subsection`) the distance between their titles is equal to the skip after the first sectioning command. Sometimes it is necessary to insert another vertical space here. To override the space inserted between sections, use the command

`\aftersectionvspace{`⟨*distance*⟩`}`

This command replaces the space inserted by a previous sectioning command with the `\vspace{`⟨*distance*⟩`}`. It works in the only case when goes right after a command producing a display section. Otherwise, the specified ⟨*distance*⟩ is ignored. The following example shows how to customize the `\subsection` command in such a way that the distance between it and a previous `\section` will be `3ex plus .5ex minus .2ex`:

```
\renewcommand\subsection{%
  \aftersectionvspace{3ex plus .5ex minus .2ex}%
  \startsection{2}}
```

`\adjustsectionmargins` Margins of a display section can be adjusted using the command

```
\adjustsectionmargins{⟨left skip⟩}{⟨right skip⟩}
```

The ⟨*left skip*⟩ and ⟨*right skip*⟩ are added to the left and right margins of the subsequent section if it is a display section. Otherwise, this command is ignored.

Modifiers. The customization of a number tag and running head of a concrete section is provided with so-call *modifiers*. A modifier is a command acting on the nearest sectioning command going after it. Usually, the modifiers are placed just before a sectioning command. All modifiers act with non-starred versions of sections. If the next sectioning command is starred, modifiers are ignored.

`\norunninghead` The `\norunninghead` modifier suppresses generation of running head for the next non-starred section, i.e. it skips the call of section mark command in the next section.

`\runninghead` The `\runninghead{`⟨*running-title*⟩`}` modifier overrides a text going to the running head when a new non-starred section starts and an appropriate `\pagestyle` is in use. This command has higher priority than the `\norunninghead`.

`\noheadingtag` The `\noheadingtag` modifier suppresses a number tag in the next section, but all other attendant actions are executed (writing to the aux-file and updating the running head).

`\headingtag` The `\headingtag{`⟨*tag*⟩`}` modifier overrides a number tag in the next section. It has the higher priority than `\noheadingtag`. Overridden section tag can be referred with the `\label` command. All fragile commands in the overridden tag should be protected.

`\headingtag*` The `\headingtag*{`⟨*tag*⟩`}` modifier prepares a number tag as is, ignoring the tag style, prefix, and suffix. The aux-file and running head are not updated in this case.

`\skipwritingtoaux` The `\skipwritingtoaux` suppresses writing to aux-file for the next section command.

NOTE: All modifiers use global settings.

`\caption`
`\caption*` The captions are implemented in this package using the same technique as the sectioning commands. There are two versions of caption command allowed within floating environments:

```
\caption[⟨toc-entry⟩]{⟨title⟩}   and
\caption*{⟨title⟩}
```

The first one works in the same manner as the standard LaTeX `\caption` command. Its starred version prepares a caption without number and preceding words 'Figure' or 'Table'.

You can use line breaking commands in captions. But in this case, you need to set the optional ⟨*toc-entry*⟩ parameter to avoid translation errors.

Caption appearance can be customized. You can customize ether all caption types or only selected caption type. The following commands do this:

```
\captionstyle[⟨type⟩]{⟨style⟩}
\captiontagstyle[⟨type⟩]{⟨style⟩}
\captiontagsuffix[⟨type⟩]{⟨suffix⟩}
\captionwidth[⟨type⟩]{⟨length⟩}
```

If ⟨*type*⟩ is omitted and these commands appear out of float environments, they are applied to all types. A command without ⟨*type*⟩ applied within a float environment is considered as a command having the type of this environment. Typed version of a command has a precedence before a non-typed one.

`\captionstyle` specifies a style the caption text will be formatted:

`default`	standard LaTeX's style,
`para`	simple paragraph without paragraph indent,
`left`	all lines are flushed left,
`center`	all lines are centered,
`right`	all lines are flushed right, or
`centerlast`	as `para`, but the last line is centered.

`\captiontagstyle` specifies a position of caption tag:

`para`	tag is formatted together with text,
`left`	tag is adjusted to the left in a separate line,
`center`	tag is centered in a separate line, or
`right`	tag is adjusted to the right in a separate line.

`\captiontagsuffix` specifies a suffix after caption tag.

`\captionwidth` specifies a width of caption.

Defaults:

```
\captionstyle{default}
\captiontagstyle{para}
\captiontagsuffix{:\hspace{0.7em plus 0.2em minus 0.1em}}
\captionwidth{\linewidth}
```

NOTE: The above-described section modifiers can be used with non-starred captions. Although, the `\runninghead` and `\norunninghead` commands have no sense with captions, but you can do them working if define a `\figuremark{}` or `\tablemark{}` command.

`\SetTOCStyle` The `\SetTOCStyle{`⟨*declarations*⟩`}` command allows customize the table of contents and other content lists. For example, the declaration

```
\SetTOCStyle{\small}
```

specifies that content lists will be prepared with the `\small` font. This command is allowed in the preamble only.

`\ChapterPrefixStyle` The appearance of Chapter/Appendix prefix in a table of contents and in a running head can be customized using the command

```
\ChapterPrefixStyle{⟨appearance list⟩}
```

The ⟨*appearance list*⟩ can contain up to two words, namely `header` and/or `toc`, delimited with a comma. Using them, you can set a prefix-style for the header and/or the table of contents, respectively. By default, the prefix-style is specified for the header only. This command is allowed for book-like classes in which the `\chapter` command is defined. It can be used in the preamble only.

3 Create New Section Styles

Along with 8 predefined section styles, you can easy create more styles.

`\newplainsectionstyle` The command

```
\newplainsectionstyle{⟨name⟩}{⟨indent⟩}[⟨pos⟩]
                     {⟨left skip⟩}{⟨right skip⟩}
```

creates a new paragraph-like section style with the given ⟨*name*⟩. It has the ⟨*indent*⟩ paragraph indent and margins specified with ⟨*left skip*⟩ and ⟨*right skip*⟩ lengths. To prepare a centered style, the optional ⟨*pos*⟩ parameter should be equal to `[c]`. In this case, left and right margins must have an additional `1fil` glue. If optional parameter is `[r]`, the left margin must have an additional `1fil` glue.

Four of predefined section styles are created using this command as follows:

```
\newplainsectionstyle{parindent}{0pt}{\parindent}{0pt}
\newplainsectionstyle{parindent*}{0pt}{\parindent}{0pt plus 1fil}
\newplainsectionstyle{center}{0pt}[c]{0pt plus 1fil}{0pt plus 1fil}
\newplainsectionstyle{centerlast}{0pt}[c]{0pt plus 1fil}{0pt plus -1fil}
```

`\newhangsectionstyle` The command

```
\newhangsectionstyle{⟨name⟩}{⟨min tag width⟩}[⟨pos⟩]
                    {⟨left skip⟩}{⟨right skip⟩}
```

creates a new hang-indented section style with the given ⟨*name*⟩. The ⟨*min tag width*⟩ length specifies a minimum width of the section tag. If a width of section tag is less than this parameter value, a white space will be inserted surround the tag to have the required width. The method of inserting a white space is the same as in the `\makebox` command. It is controlled with the optional ⟨*pos*⟩ parameter (`l`, `c`, or `r`; `l` default). Other parameters have the same meaning as in the previous command.

Four of predefined section styles are created using this command as follows:

```
\newhangsectionstyle{hangindent}{0pt}{0pt}{0pt}
\newhangsectionstyle{hangindent*}{0pt}{0pt}{0pt plus 1fil}
\newhangsectionstyle{hangparindent}{0pt}{\parindent}{0pt}
\newhangsectionstyle{hangparindent*}{0pt}{\parindent}{0pt plus 1fil}
```

The following examples shows possibilities of these commands:

3.1 This subsection was prepared in the margin style

The definition of the `margin` style is the following:

```
\newhangsectionstyle{margin}{2in}[r]{-2in}{0pt plus 1fil}
```

3.2 This subsection was prepared in the list style

The definition of the `list` style is the following:

```
\newhangsectionstyle{list}{1in}{0pt}{1in plus 1fil}
```

3.3 This subsection was prepared in the flushright style

The definition of the `flushright` style is the following:

```
\newplainsectionstyle{flushright}{0pt}[r]{1in plus 1fil}{0pt}
```

4 Declare Sections and Captions

`\DeclareSection` To define or redefine a section or caption command, you can use in the preamble of your document the following command:

`\DeclareSection{`⟨*level*⟩`}{`⟨*type*⟩`}[`⟨*indent*⟩`]{`⟨*prefix*⟩`}{`⟨*beforeskip*⟩`}`
`{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`

⟨*level*⟩ a section level number. Zero and negative values are interpreted as follows: 0 means declaring the `\chapter` or `\part` command depending on a class used; a negative value means declaring a caption.

⟨*type*⟩ a section type. For zero level, this parameter is ignored. For negative level, it defines a float type (i.e., `figure` or `table`). For positive level, it defines a counter name. The name of marking command is composed from the type as `\`⟨*type*⟩`mark`.

⟨*indent*⟩ indentation of heading from the left margin (zero is default). Ignored for negative levels.

⟨*prefix*⟩ a prefix inserted before a section-number tag (usually empty). In chapter, part, or caption declaration commands, it is inserted right before the tag name, e.g., before the `\@chapapp`, `\partname`, `\figurename`, or `\tablename` command.

⟨*beforeskip*⟩ the skip to leave above the heading.

⟨*afterskip*⟩ if positive, then the skip to leave below the heading, else negative of skip to leave to right of running heading. The negative value is allowed for positive section levels only.

⟨*style*⟩ commands to set a style. The last command in this argument may be a command such as `\MakeUppercase` that takes an argument. The section heading will be supplied as the argument to this command. So setting it to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

Sections having nonnegative ⟨*level*⟩ and positive ⟨*afterskip*⟩ are display sections. They are declared with the `hangindent` style and do not obey the `\sectionstyle` command.

`\DeclareSection*` To declare a display section having dynamic alignment controlled with the `\sectionstyle` command, use the star-version of the previous command:

```
\DeclareSection*{⟨level⟩}{⟨type⟩}{⟨prefix⟩}{⟨beforeskip⟩}
                {⟨afterskip⟩}{⟨style⟩}
```

A negative ⟨*afterskip*⟩ has no meaning in this case.

`\bff` To prepare bold section headings, you can use the `\bff` command in the ⟨*style*⟩ parameter. It tries to set everything bold. Its definition is the following:

```
\newcommand{\bff}{\normalfont\bfseries\mathversion{bold}}
```

Examples of section and caption declarations:

```
\DeclareSection{-2}{table}{}{0pt}{10pt}{}
\DeclareSection{-1}{figure}{}{10pt}{0pt}{}
\DeclareSection*{1}{section}{}%
                {3.5ex plus 1ex minus .2ex}%
                {2.3ex plus .2ex}{\Large\bff}
```

Here we declare the table caption command with zero skip before it and `10pt` skip after it. On contrary, the figure caption command produces `10pt` skip before it and zero skip after it. The `\section` command is declared with dynamic horizontal alignment. It is prepared in the `\Large` font with everything bold.

`\SectionTagSuffix` The `\SectionTagSuffix{⟨suffix⟩}` command specifies a default suffix inserted after a section number tag. For example, the command

```
\SectionTagSuffix{.\quad}
```

sets the decimal point after every section number tag. Sections of 0th level ignore this suffix. The default tag is `\quad`. The command can be used in the preamble only.

`\RunningSectionSuffix` The `\RunningSectionSuffix{`⟨*suffix*⟩`}` command specifies a suffix inserted after a running section title right before the skip after section. It can be used in the preamble only. The default value is an empty suffix.

`\norunningsuffix` To remove the suffix after a running section, put the `\norunningsuffix` modifier in the parameter of running section.

`\CaptionTagSuffix` The `\CaptionTagSuffix{`⟨*suffix*⟩`}` command specifies a default suffix inserted after a caption number tag. It can be used in the preamble only. The default caption tag is:

```
\CaptionTagSuffix{:\hspace{0.7em plus 0.2em minus 0.1em}}
```

5 Declare TOC-Entries

`\DeclareTOCEntry` To declare an entry of table of contents or other lists (list of figures or list of tables), use the following command (in the preamble only):

`\DeclareTOCEntry{`⟨*level*⟩`}{`⟨*action*⟩`}{`⟨*prefix*⟩`}{`⟨*prototype*⟩`}`
`{`⟨*style*⟩`}[`⟨*next*⟩`]`

⟨*level*⟩ a section level number. For zero and negative level the following commands are created: 0 means `\l@chapter` or `\l@part` depending on class used; −1 means `\l@figure`; −2 means `\l@table`. If level is greater than 5, the name of toc-entry command is generated as `\l@section@`⟨*level-in-roman*⟩, i.e., the toc-entry of 6th level is `\l@section@vi`.

⟨*action*⟩ commands applied before entry is produced (usually empty).

⟨*prefix*⟩ text inserted before the section number (usually empty).

⟨*prototype*⟩ prototype of number for alignment the toc-entry body. The hang indent of this toc-entry will be equal to the width of

⟨*style*⟩`{`⟨*prefix*⟩⟨*prototype*⟩⟨*numberline-suffix*⟩`}`

⟨*style*⟩ commands to set a style. The last command in this argument may be a command such as `\MakeUppercase` that takes an argument. The produced entry will be supplied as the argument to this command. So setting it to, say, `\bfseries\MakeUppercase` would produce bold, uppercase entry. This style is applied to the number also and to the page number. To apply different styles to the text of entry and to its page number, use in this parameter the command

`\applystyle{`⟨*text-style*⟩`}{`⟨*number-style*⟩`}`

⟨*next*⟩	prototype for left margin adjustment for an entry of the next level. Default is the hang indent of the current toc-entry.

A toc-entry is produced within a group.

`\NumberlineSuffix` The `\NumberlineSuffix{`⟨*calc-suffix*⟩`}{`⟨*actual-suffix*⟩`}` command allows customize a skip inserted after numbers in TOC-like entries. The ⟨*calc-suffix*⟩ parameter is used in calculations of hang indent of toc-entries and the ⟨*actual-suffix*⟩ is really inserted at the end of number. The `{`⟨*calc-suffix*⟩`}` is usually wider than the ⟨*actual-suffix*⟩. The default is `\NumberlineSuffix{\quad}{\enskip}`. This command is available in the preamble only.

`\PnumPrototype` The `\PnumPrototype{`⟨*prototype*⟩`}` command is used for adjustment the right margin of the text of toc-entries in toc-lists. Default is `\PnumPrototype{99}`. If your document has more than 99 pages, use `\PnumPrototype{999}`. This command is available in the preamble only.

`\TOCMarginDrift` The `\TOCMarginDrift{`⟨*increment*⟩`}` command specifies a value of right-margin drift in TOCs. The increment is applied after the `\@plus` token in definition of right margin. Empty argument means no drift. Examples:

```
\TOCMarginDrift{2em}
\TOCMarginDrift{1fil}
```

The command can be use anywhere in the document.

`\runinsectionskip` This command is useful in the ⟨*action*⟩ parameter of the toc-entry declaration to produce the skip before a toc-entry equal to the skip before run-in sections.

The following example shows how toc-entries are declared in books:

```
\DeclareTOCEntry{-2}{}{}{9.9}{}% table
\DeclareTOCEntry{-1}{}{}{9.9}{}% figure
\DeclareTOCEntry{0}{\runinsectionskip\def\@dotsep{1000}%
  \aftergroup\penalty\aftergroup\@highpenalty}{}{9}{\bff}% chapter
\DeclareTOCEntry{1}{}{}{9.9}{}[9.9]% section
\DeclareTOCEntry{2}{}{}{9.9.9}{}[9.9.9]% subsection
\DeclareTOCEntry{3}{}{}{}{}[\qquad]% subsubsection
```

The number prototype for figures and tables is '9.9' here. The `\l@chapter` entry applies the run-in section skip before it and redefines the `\@dotsep` command to remove dot leaders. Using the `\aftergroup` command, it inserts the `\@highpenalty` after this toc-entry to avoid a page break at this point. The left margin adjustment after section and nested toc-entries is calculated here using the prototype of widest section number. This produces the following nesting:

1 Chapter
  1.1 Section
    1.1.1 Subsection
        Subsubsection

6 Declare New Float Types

The standard LaTeX classes provide two types of floating environments: figures and tables. If you have prepared a new floating environment in some way (i.e., using the `float` package by Anselm Lingnau), you can declare a caption for the new float with the commands described in previous sections.

`\RegisterFloatType` In books, when a new chapter starts, the `\chapter` command puts a special vertical skip to the contents of list of figures and of list of tables. This behaviour can be easy extended to new float types if you register them within this package. The registration is provided with the following command:

```
\RegisterFloatType{⟨float-type⟩}
```

After the float type is registered, you can declare a toc-entry for it using the negation of its registration number in the ⟨*level*⟩ parameter. The first new float type is registered third (after the figure and table). So, you must use ⟨*level*⟩ $= -3$ for it, -4 for the next registered float type and so on.

In the following example, we define a new float type, `program`, and prepare the caption and toc-entry commands for it. The caption of programs is supposed to be used at the beginning of program. So, we make it in the same manner as the table caption.

```
\documentclass{book}
\usepackage{float,nccsect}
\newfloat{program}{tp}{lop}[chapter]
\floatname{program}{Program}
\RegisterFloatType{program}
\DeclareSection{-3}{program}{}{0pt}{10pt}{}
\DeclareTOCEntry{-3}{}{}{9.9}{}
```

To produce a list of programs, you can then use the `\listof` command from the `float` package as follows:

```
\listof{program}{List of Programs}
```

7 Epigraphs and Related Staff

`\beforechapter` `\epigraph` To put epigraph before any chapter, you can use two methods: low-level `\beforechapter{`⟨*anything*⟩`}` hook or user-level command

```
\epigraph[⟨width⟩]{⟨text⟩}{⟨author⟩}
```

The last one applies a special formatting to epigraph and calls the first one. The `\beforechapter` hook inserts its contents at the beginning of page just before a chapter instead of spacing specified in the chapter declaration.

`\epigraphparameters` Formatting of user-level epigraph is provided with the following command

```
\epigraphparameters{⟨style⟩}{⟨width⟩}{⟨height⟩}{⟨author-style⟩}
                    {⟨after-action⟩}
```

Here ⟨*style*⟩ is a style applied to the whole epigraph (font selection, spacing and positioning, etc.), the ⟨*width*⟩ is the default epigraph width (can be changed in an epigraph), the ⟨*author-style*⟩ is the style applied to the author's signature, and the ⟨*after-action*⟩ is an action applied after the epigraph (usually a vertical spacing). All styles and actions are applied in the vertical mode. An ⟨*author-style*⟩ can finish with one-argument macro getting the author of epigraph and formatting it.

`\epigraphwidth` In `\epigraphparameters`, you can use the `\epigraphwidth` macro which contains a selected epigraph width.

The default style is:

```
\epigraphparameters{\StartFromHeaderArea\small\raggedleft}
                   {.45\linewidth}{5\baselineskip}
                   {\raggedleft\itshape}{\vspace{2ex}}
```

`\StartFromTextArea` The `\vspace*` command applied at the beginning of page has one serious disadvantage: it skips more space that specified in its parameter. To remove this disadvantage, we introduce the `\StartFromTextArea` command that inserts a zero-height strut and allows use the `\vspace` command after it without troubles.

`\StartFromHeaderArea` You can also extend the text area on the header if apply the `\StartFromHeaderArea` command at the beginning of page. Such action is useful in epigraphs: the first chapter's page usually has an empty header and positioning an epigraph from the header is the good practice.

8 Declare Part

The `\part` command in book-like classes is the only sectioning command that cannot be prepared with the `\DeclareSection` command. So, we add special declarations to provide parts in books with features of other sectioning commands.

`\DeclarePart` To redefine the `\part` in books, use the following declaration:

`\DeclarePart{`⟨*before*⟩`}{`⟨*after*⟩`}{`⟨*prefix*⟩`}{`⟨*style*⟩`}`

⟨*before*⟩ an action applied before a part at the beginning of page. It usually specifies a vertical skip `\vfil` and a paragraph style to be applied to the part number tag and title.

⟨*after*⟩ an action applied after the part. It usually contains `\vfil` and page finishing commands.

⟨*prefix*⟩ a prefix inserted before a part tag. It contains style commands to be applied to the tag and the `\vspace` command specifying a distance between the part tag and title. The `\partname` command goes right after the prefix.

⟨*style*⟩ a style to be applied to the part title. It can end with `\MakeUppercase`.

The default declaration of the `\part` is the following:

```
\DeclarePart{\StartFromTextArea\vfil\centering}%
            {\vfil\newpage \if@twoside\if@openright
               \mbox{}\thispagestyle{empty}\newpage\fi\fi}%
            {\vspace{4ex}\huge\bff}{\Huge\bff}
```

The `\StartFromTextArea` command prevents ignoring a vertical space at the beginning of page. All paragraphs of part title are centered horizontally using the `\centering` declaration, and the title is centered vertically using `\vfil` commands before and after it. A page after the part is made empty in two-side mode if it is even. The space after the part tag is set to `4ex`.

In Russian typesetting tradition, the part can be prepared in the same manner as a chapter, i.e. a text going after a part is prepared on the same page with the part title. It is easy to re-declare the part in such style. Let us start a part from the header and delimit it from the text with a decorative line. The following declaration does this:

```
\DeclarePart{\StartFromHeaderArea\centering}
            {\vspace{2mm}\noindent\hrulefill\par
             \addvspace{5mm}}
            {\vspace{.5em}\LARGE\bff}{\Huge\bff}
```

But when a chapter goes right after a part, we need to place the part and chapter titles together on the same page. This can be applied using the `\beforechapter` hook:

```
\beforechapter{\part{⟨part title⟩}}
\chapter{⟨chapter title⟩}
```

Modifiers stored in the parameter of `\beforechapter` hook will act on the `\part` command. Modifiers outside of `\beforechapter` will act on the `\chapter` command.

`\DeclareTOCPart` To produce a toc-entry command for a part, the following declaration is specified for book-like classes:

```
\DeclareTOCPart{⟨action⟩}[⟨afterskip⟩]{⟨prefix⟩}{⟨prototype⟩}{⟨style⟩}
```

⟨*action*⟩ an action applied before the part toc-entry. It usually a skip before part. It is recommended to prepare it with `\NCC@secskip` command.

⟨*afterskip*⟩ a skip after this entry. If it is omitted, the default `\NCC@runskip` value is applied after this entry.

⟨*prefix*⟩ a prefix inserted before a part tag (usually empty).

⟨*prototype*⟩ a prototype of part tag used for calculation the hang indent in this entry.

⟨*style*⟩ a style applied to the whole text of entry and to the page number. The `\MakeUppercase` is allowed to finish this parameter. The `\applystyle` command can be used inside it to apply different styles to the toc-entry and the page number.

The default declaration of the part toc-entry is the following:

```
\DeclareTOCPart{\NCC@secskip{4ex \@plus .2ex}%
                \def\@dotsep{1000}}%
               {}{\partname\ II}{\large\bff}
```

9 The Implementation

The **afterpackage** package is used to add compatibility commands.

```
⟨*package⟩
\RequirePackage{afterpackage}
```

`\NCC@secskip` `\NCC@runskip` The package shares the following commands with the `nccthm` package:

> `\NCC@secskip{`⟨*skip*⟩`}` adds the ⟨*skip*⟩ before a section,
> `\NCC@runskip` is a skip inserted before run-in sections.

We protect the definitions of these commands with testing the `nccthm` package to be already loaded.

```
\@ifpackageloaded{nccthm}{}{%
  \def\NCC@secskip#1{%
    \if@noskipsec \leavevmode \fi \par
    \if@nobreak \everypar{}\else
      \addpenalty\@secpenalty
      \addvspace{#1}%
    \fi
  }
  \def\NCC@runskip{2.75ex \@plus 1ex \@minus .2ex}
}
```

`\runinsectionskip` This command is useful in toc-entries:

```
\newcommand{\runinsectionskip}{\NCC@secskip{\NCC@runskip}}
```

9.1 The Kernel

We start with declaring the section controls (modifiers):

`NCC@nosectag`	is true if `\noheadingtag` is applied;
`NCC@secstartag`	is true if `\headingtag*{`⟨*tag*⟩`}` is applied;
`\NCC@sectag`	saves a value of the `\headingtag` parameter;
`NCC@nosecmark`	is true if `\norunninghead` is applied;
`\NCC@secmark{`⟨*mark-command*⟩`}`	executes the ⟨*mark-command*⟩ with the parameter of `\runninghead` command;
`NCC@noaux`	is true if `\skipwritingtoaux` is applied.

```
\newif\ifNCC@nosectag
\newif\ifNCC@secstartag
\newif\ifNCC@nosecmark
\newif\ifNCC@noaux
```

`\NCC@global` We reset all controls globally, but in the `\beforechapter` hook we need to reset them locally. So, we reset all controls using the `\NCC@global` modifier which value is `\global` by default.

```
\let\NCC@global\global
```

`\NCC@sec@reset@controls` This command resets all controls to default values. It must be applied at the end of every section command.

```
\def\NCC@sec@reset@controls{%
  \NCC@global\NCC@nosectagfalse
  \NCC@global\NCC@secstartagfalse
  \NCC@global\let\NCC@sectag\relax
  \NCC@global\NCC@nosecmarkfalse
  \NCC@global\let\NCC@secmark\relax
  \NCC@global\NCC@noauxfalse
}
\NCC@sec@reset@controls
```

`\norunninghead` `\runninghead` `\noheadingtag` `\headingtag` `\headingtag*` `\skipwritingtoaux` User interface to section controls:

```
\newcommand{\norunninghead}{\NCC@global\NCC@nosecmarktrue}
\newcommand*{\runninghead}[1]{\NCC@global\def\NCC@secmark##1{##1{#1}}}
\newcommand{\noheadingtag}{\NCC@global\NCC@nosectagtrue}
\newcommand{\headingtag}{%
  \@ifstar{\NCC@global\NCC@secstartagtrue\NCC@setsectag}{\NCC@setsectag}%
}
\def\NCC@setsectag#1{\NCC@global\def\NCC@sectag{#1}}
\newcommand{\skipwritingtoaux}{\NCC@global\NCC@noauxtrue}
```

`\NCC@makesection` The `\NCC@makesection{`⟨*type*⟩`}{`⟨*level*⟩`}{`⟨*toc-entry*⟩`}{`⟨*toc-action*⟩`}` produces a section or caption. It analyzes the modifiers and customizes sections or captions. The ⟨*toc-action*⟩ parameter contains an attendant action that must be applied at the end of macro. It writes a toc-entry to aux-file.

The command uses the following hooks that must be prepared before its call:

`\NCC@makesectag{`⟨*value*⟩`}` produces a tag in `\@svsec` using the given value;

`\NCC@make{`⟨*action*⟩`}` makes a section or caption heading and applies the ⟨*action*⟩ after heading. Before the call of this command, the `\@svsec` macro is prepared (it contains a prepared tag).

We start from the case when the `\headingtag*{`⟨*tag*⟩`}` modifier was applied and the section tag was saved in the `\NCC@sectag` macro. We apply the `\NCC@make` procedure with the given section tag. Attendant actions are ignored for this case.

```
\def\NCC@makesection#1#2#3#4{%
  \ifNCC@secstartag
  \let\@svsec\NCC@sectag
  \NCC@make{}%
\else
```

Prepare a tag creation command in the `\the`⟨*type*⟩ macro. We can do some temporary changes here that will be restored at the end of macro. The restore hook is prepared in the `\NCC@restsec` command.

```
  \ifx\NCC@sectag\relax
```

The `\noheadingtag` case: we temporary set the `secnumdepth` counter to very low negative value. This prevents numbering this section:

```
    \ifNCC@nosectag
      \edef\NCC@restsec{%
        \noexpand\c@secnumdepth \the\c@secnumdepth\relax
      }%
      \c@secnumdepth -1000
```

The ordinary case: No restore actions is necessary here.

```
    \else
      \let\NCC@restsec\relax
      \ifnum#2>\c@secnumdepth \else\refstepcounter{#1}\fi
    \fi
```

The `\headingtag{`⟨*tag*⟩`}` case: we temporary let the `\the`⟨*type*⟩ macro to be equal to the `\NCC@sectag` command produced by the `\headingtag`, save the original value in the `\NCC@thesec` command, and prepare the `\NCC@restsec` macro.

```
  \else
    \expandafter\let\expandafter\NCC@thesec\csname the#1\endcsname
    \def\NCC@restsec{%
      \expandafter\let\csname the#1\endcsname\NCC@thesec
    }%
    \expandafter\let\csname the#1\endcsname\NCC@sectag
    \protected@edef\@currentlabel{\NCC@sectag}%
  \fi
```

Prepare section tag in the `\@svsec` command:

```
  \ifnum #2>\c@secnumdepth
    \let\@svsec\@empty
  \else
    \protected@edef\@svsec{%
      \protect\NCC@makesectag{\csname the#1\endcsname}%
    }%
  \fi
```

We cannot do the marking right now because the producing of section can be suspended to the beginning of the nearest paragraph (in run-in sections). So, we need to prepare a mark action in a command that will save its state as long as necessary. This command is `\NCC@makemark`.

```
  \let\NCC@makemark\@empty
  \@ifundefined{#1mark}{}{%
    \ifx\NCC@secmark\relax
```

Ordinary case: prepare the section mark with the ⟨*toc-entry*⟩ parameter.

```
        \ifNCC@nosecmark \else
          \def\NCC@makemark{\csname #1mark\endcsname{#3}}%
        \fi
```

The `\runninghead{`⟨*heading*⟩`}` case: pass the mark command in the parameter of `\NCC@secmark`. We need to save the `\NCC@secmark` value in some command and pass this command within `\NCC@makemark` because the `\NCC@secmark` could be removed before the use.

```
      \else
        \let\NCC@savesecmark\NCC@secmark
        \def\NCC@makemark{%
          \NCC@savesecmark{\csname #1mark\endcsname}%
          \let\NCC@savesecmark\relax
        }%
      \fi
    }%
```

Make the section. We must apply the restore action at the end action of `\NCC@make` command by the same reason that the section making may be suspended:

```
    \ifNCC@noaux
      \NCC@make{\NCC@makemark \NCC@restsec}%
    \else
      \NCC@make{\NCC@makemark #4\NCC@restsec}%
    \fi
```

Reset modifiers:

```
  \fi
  \NCC@sec@reset@controls
}
```

9.2 Section Making Commands

`\indentaftersection` `\noindentaftersection` Introduce macros controlling indentation after display sections:

```
\newcommand{\indentaftersection}{\@afterindenttrue}
\newcommand{\noindentaftersection}{\@afterindentfalse}
```

`\SectionTagSuffix` The `\SectionTagSuffix{`⟨*suffix*⟩`}` sets a default suffix after a section tag.

```
\newcommand*{\SectionTagSuffix}[1]{\def\NCC@asecnum{#1}}
\@onlypreamble\SectionTagSuffix
```

`\sectiontagsuffix` `\sectiontagsuffix[`⟨*type*⟩`]{`⟨*suffix*⟩`}` changes a suffix after section tag that will be used for sections of the given ⟨*type*⟩. If ⟨*type*⟩ is omitted, the specified suffix will be used in text flow for all sections having no special suffix.

```
\newcommand*\sectiontagsuffix[2][]{%
  \expandafter\def\csname NCC@asecnum@#1\endcsname{\def\NCC@asecnum{#2}}%
}
\let\NCC@asecnum@\@empty
```

```
\def\NCC@setsectionsuffix#1{%
  \edef\@tempa{NCC@asecnum@\NCC@secname{#1}}%
  \@ifundefined{\@tempa}{%
    \let\NCC@asecnumset\NCC@asecnum@
  }{%
    \expandafter\let\expandafter\NCC@asecnumset\csname\@tempa\endcsname
  }%
}
```

`\RunningSectionSuffix` The `\RunningSectionSuffix{`⟨*suffix*⟩`}` sets a suffix after a title of a running section:

```
\newcommand*{\RunningSectionSuffix}[1]{\def\NCC@asectitle{\unskip#1}}
\@onlypreamble\RunningSectionSuffix
```

`\NCC@preparesectag` The `\NCC@preparesectag{`⟨*style*⟩`}{`⟨*before*⟩`}` hook prepares the `\NCC@makesectag` command:

```
\def\NCC@preparesectag#1#2{\def\NCC@makesectag##1{#1#2##1\NCC@asecnum}}
```

`\NCC@secname` The `\NCC@secname{`⟨*level*⟩`}` command generates section name (`main`, `section`, `subsection`, ..., or `section@vi`, `section@vii`, ... for new section levels). This name is used as the second parameter of the `\addcontentsline` command, in the declarations of toc-entries, and in the style selection command.

```
\def\NCC@secname#1{%
  \ifcase#1main\or section\or subsection\or subsubsection\or
    paragraph\or subparagraph\else section@\romannumeral#1\fi
}
```

`\NCC@startsection` The `\NCC@startsection` command has the same syntax as its non-NCC prototype:

> `\NCC@startsection{`⟨*type*⟩`}{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*beforeskip*⟩`}`
> `{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`

but works in a bit different way: it ignores the sign of ⟨*beforeskip*⟩. In the original version the testing was applied to set an appropriate `afterindent` mode. But we change this mode using `\indentaftersection` and `\noindentaftersection` macros.

```
\def\NCC@startsection#1#2#3#4#5#6{%
  \@tempskipa #4\relax
  \ifdim \@tempskipa <\z@ \@temskipa -\@tempskipa \fi
  \NCC@secskip \@tempskipa
  \secdef{\NCC@sect{#1}{#2}{#3}{#4}{#5}{#6}}{\NCC@ssect{#3}{#4}{#5}{#6}}%
}
```

`\NCC@makesec` The interface of `\NCC@ssect` and `\NCC@sect` commands is similar to their LaTeX's prototypes. They are based on the following command:

> `\NCC@makesec{`⟨*indent*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}{`⟨*afterskip*⟩`}{`⟨*action*⟩`}`

In fact, there are two versions of this command: the traditional version, `\NCC@makesect`, and the version with dynamic control of section style, `\NCC@makesecx`. One of them should be selected before applying the `\NCC@ssect` and `\NCC@sect` commands.

`\NCC@ssect` The starred form of section:

> `\NCC@ssect{`⟨*indent*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`

```
\def\NCC@ssect#1#2#3#4#5{%
  \let\@svsec\@empty
  \NCC@makesec{#1}{#4}{#5}{#3}{}%
  \NCC@sec@reset@controls
}
```

`\NCC@sect` The base form of section:

> `\NCC@sect{`⟨*type*⟩`}{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`
> `[`⟨*toc-entry*⟩`]{`⟨*heading*⟩`}`

```
\def\NCC@sect#1#2#3#4#5#6[#7]#8{%
  \def\NCC@make{\NCC@makesec{#3}{#6}{#8}{#5}}%
  \NCC@makesection{#1}{#2}{#7}{%
    \addcontentsline{toc}{\NCC@secname{#2}}{%
      \ifnum #2>\c@secnumdepth \else
        \numberline{\csname the#1\endcsname}%
      \fi
      #7%
    }%
  }%
}
```

`\NCC@makesect` The traditional section making command:

> `\NCC@makesect{`⟨*indent*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}{`⟨*afterskip*⟩`}{`⟨*action*⟩`}`

```
\def\NCC@makesect#1#2#3#4#5{%
  \@tempskipa #4\relax
  \ifdim \@tempskipa>\z@
    \begingroup \normalfont
      \NCC@asecnumset
```

The `\NCC@secttitle{`⟨*style*⟩`}{`⟨*tag*⟩`}{`⟨*title*⟩`}` hook prepares traditional display section:

```
      \NCC@secttitle{#2}{\NCC@hangfrom{\hskip #1\relax\@svsec}}%
        {\interlinepenalty \@M\ignorespaces #3\@@par}
    \endgroup
    #5%
  \else
```

The `\NCC@secptitle{`⟨*style*⟩`}{`⟨*tag*⟩`}{`⟨*title*⟩`}` hook prepares running section. The `\norunningsuffix` modifier applied in the parameter of running section removes a suffix after section title.

```
    \def\@svsechd{{\normalfont
      \NCC@asecnumset
      \def\norunningsuffix{\protect\NCC@nosecsuffix}%
      \NCC@secptitle{#2}{\hskip #1\relax{\@svsec}}%
        {\ignorespaces #3\NCC@asectitle}#5}}%
  \fi
  \@xsect{#4}%
}
\def\NCC@secttitle#1#2#3{#1{#2#3}}
\def\NCC@secptitle#1#2#3{#1{#2#3}}
\newcommand*\norunningsuffix{}
\def\NCC@nosecsuffix{\let\NCC@asectitle\@empty}
```

9.3 Create Section Styles

`\NCC@hangfrom`

`\NCC@hangfrom{`⟨*section tag*⟩`}` works as the LaTeX's `\@hangfrom` command, but its margins can be adjusted with the `\adjustsectionmargins` command.

```
\def\NCC@hangfrom{%
  \NCC@setsecmargins{\z@skip}{\z@skip}\NCC@hangsecstyle{\z@}{}%
}
```

`\NCC@setsecmargins`

`\NCC@setsecmargins{`⟨*left skip*⟩`}{`⟨*right skip*⟩`}` sets section margins and applies the hook that can be defined by the `\adjustsectionmargins` command.

```
\def\NCC@setsecmargins#1#2{%
  \leftskip\z@skip \rightskip\z@skip
  \parfillskip\@flushglue
  \let\\\@normalcr
  \NCC@adjsecmargins{#1}{#2}%
  \NCC@secmarginshook
}
```

`\NCC@adjsecmargins`

`\NCC@adjsecmargins{`⟨*left skip*⟩`}{`⟨*right skip*⟩`}` adjusts section margins. The `\parfillskip` value is also adjusted to a difference between stretchabilities of the ⟨*left skip*⟩ and the ⟨*right skip*⟩. Using this trick, we can easy specify the `centerlast` style just setting the stretchability of the ⟨*right skip*⟩ as a negation of the ⟨*left skip*⟩ stretchability. To extract a stretchability from a skip, we simply add it multiplied by `-1` (while multiplication the stretchability is removed!).

```
\def\NCC@adjsecmargins#1#2{%
  \setlength\@tempskipa{#1}\advance\leftskip\@tempskipa
  \setlength\@tempskipb{#2}\advance\rightskip\@tempskipb
  \advance\@tempskipa -1\@tempskipa \advance\@tempskipb -1\@tempskipb
  \advance\@tempskipa -\@tempskipb \advance\parfillskip\@tempskipa
}
```

`\NCC@hangsecstyle`

`\NCC@hangsecstyle{`⟨*min tag width*⟩`}{`⟨*pos*⟩`}{`⟨*section tag*⟩`}` starts a hang paragraph and prints its tag. The ⟨*min tag width*⟩ specifies a minimum width of hang indent and ⟨*pos*⟩ specifies an alignment of ⟨*section tag*⟩ (`l`, `c`, `r`) if its width is less than the minimum width.

```
\def\NCC@hangsecstyle#1#2#3{%
  \setlength\@tempdima{#1}%
  \setbox\@tempboxa\hbox{#3}%
  \ifdim \wd\@tempboxa > \@tempdima
    \hangindent\wd\@tempboxa \noindent \box\@tempboxa
  \else
    \hangindent\@tempdima
    \noindent \makebox[\@tempdima][#2]{\unhbox\@tempboxa}%
  \fi
}
```

`\adjustsectionmargins` `\adjustsectionmargins{`⟨*left skip*⟩`}{`⟨*right skip*⟩`}` defines the `\NCC@secmarginshook` macro to be applied after margins are set. To be sure this hook will be applied only once, we release it in the `\NCC@sec@reset@controls` hook.

```
\newcommand*\adjustsectionmargins[2]{%
  \NCC@global\def\NCC@secmarginshook{\NCC@adjsecmargins{#1}{#2}}
}
\g@addto@macro\NCC@sec@reset@controls{%
  \NCC@global\let\NCC@secmarginshook\@empty
}
\let\NCC@secmarginshook\@empty
```

`\NCC@sec` A style of sections having dynamic control is defined by the `\NCC@sec{`⟨*tag*⟩`}` hook. This hook is applied inside a group preparing a heading. All section style commands redefine this hook.

`\newplainsectionstyle` `\newplainsectionstyle{`⟨*name*⟩`}{`⟨*indent*⟩`}[`⟨*pos*⟩`]{`⟨*left skip*⟩`}{`⟨*right skip*⟩`}`

```
\newcommand*\newplainsectionstyle[2]{%
  \@ifnextchar[{\NCC@newplainsec{#1}{#2}}{\NCC@newplainsec{#1}{#2}[l]}%
}
\def\NCC@newplainsec#1#2[#3]#4#5{%
  \def\@tempa{#3}\def\@tempb{c}%
  \ifx\@tempa\@tempb
    \expandafter\newcommand\csname NCC@sec@#1\endcsname
      {\def\NCC@sec{\NCC@setsecmargins{#4}{#5}%
        \let\\\@centercr \advance\parfillskip -\@flushglue
        \setlength\parindent{#2}}}%
  \else
    \def\@tempb{r}%
    \ifx\@tempa\@tempb
      \expandafter\newcommand\csname NCC@sec@#1\endcsname
        {\def\NCC@sec{\NCC@setsecmargins{#4}{#5}%
          \let\\\@centercr \advance\parfillskip -\@flushglue
          \advance\parfillskip -\@flushglue
          \setlength\parindent{#2}}}%
    \else
      \expandafter\newcommand\csname NCC@sec@#1\endcsname
        {\def\NCC@sec{\NCC@setsecmargins{#4}{#5}\setlength\parindent{#2}}}%
    \fi
  \fi
}
\@onlypreamble\newplainsectionstyle
\@onlypreamble\NCC@newplainsec
```

`\newhangsectionstyle` `\newhangsectionstyle{`⟨*name*⟩`}{`⟨*min tag width*⟩`}[`⟨*pos*⟩`]{`⟨*left skip*⟩`}{`⟨*right skip*⟩`}`

```
\newcommand*\newhangsectionstyle[2]{%
  \@ifnextchar[{\NCC@newhangsec{#1}{#2}}{\NCC@newhangsec{#1}{#2}[l]}%
}
\def\NCC@newhangsec#1#2[#3]#4#5{%
  \expandafter\newcommand\csname NCC@sec@#1\endcsname
    {\def\NCC@sec{\NCC@setsecmargins{#4}{#5}\NCC@hangsecstyle{#2}{#3}}}%
}
\@onlypreamble\newhangsectionstyle
\@onlypreamble\NCC@newhangsec
```

Specify predefined section styles. The `\@flushglue` is equal to `0pt plus 1fil`.

```
\newhangsectionstyle{hangindent}{\z@}{\z@skip}{\z@skip}
\newhangsectionstyle{hangindent*}{\z@}{\z@skip}{\@flushglue}
\newhangsectionstyle{hangparindent}{\z@}{\parindent}{\z@skip}
\newhangsectionstyle{hangparindent*}{\z@}{\parindent}{\@flushglue}
\newplainsectionstyle{parindent}{\z@}{\parindent}{\z@skip}
\newplainsectionstyle{parindent*}{\z@}{\parindent}{\@flushglue}
\newplainsectionstyle{center}{\z@}[c]{\@flushglue}{\@flushglue}
\newplainsectionstyle{centerlast}{\z@}[c]{\@flushglue}{-\@flushglue}
```

9.4 Make Sections with Dynamic Control

`\sectionstyle` The `\sectionstyle[`⟨*type*⟩`]{`⟨*style*⟩`}` changes a style for display sections of the given ⟨*type*⟩.

```
\newcommand*{\sectionstyle}[2][]{%
  \@ifundefined{NCC@sec@#2}{%
    \PackageError{nccsect}{Unknown section style `#2'}{}%
  }{%
    \expandafter\def\csname NCC@secstyle@#1\endcsname{%
      \csname NCC@sec@#2\endcsname
    }%
  }%
}
```

`\NCC@setsectionstyle` The `\NCC@setsectionstyle{`⟨*level*⟩`}` set a style for the given section level. If a style for the given level is undefined, the default style is selected.

```
\def\NCC@setsectionstyle#1{%
  \edef\@tempa{NCC@secstyle@\NCC@secname{#1}}%
  \@ifundefined{\@tempa}{\NCC@secstyle@}{\csname\@tempa\endcsname}%
}
```

`\NCC@makesecx` The dynamic section making command:

\NCC@makesecx{⟨*indent*⟩}{⟨*style*⟩}{⟨*heading*⟩}{⟨*afterskip*⟩}{⟨*action*⟩}

It prepares only display sections and ignores the ⟨*indent*⟩ parameter.

```
\def\NCC@makesecx#1#2#3#4#5{%
  \begingroup\normalfont
    \NCC@asecnumset
```

The \NCC@secxtitle{⟨*style*⟩}{⟨*tag*⟩}{⟨*title*⟩} hook prepares display section with dynamic control. The \NCC@sec macro is protected to prevent its expansion by \MakeUppercase.

```
    \NCC@secxtitle{#2}{\protect\NCC@sec{\@svsec}}
      {\interlinepenalty \@M\ignorespaces #3\@@par}%
  \endgroup #5%
  \par \nobreak \vskip #4\relax \@afterheading \ignorespaces
}
\def\NCC@secxtitle#1#2#3{#1{#2#3}}
```

9.5 Make the Main Section

\partmark Define the \partmark if it is undefined yet.

```
\providecommand*\partmark[1]{\markboth{}{}}
```

\NCC@startmainsec The main section is a section of zero level. It is prepared with the following command:

\NCC@startmainsec{⟨*alignment*⟩}{⟨*prefix*⟩}{⟨*beforeskip*⟩}
{⟨*afterskip*⟩}{⟨*style*⟩}

It starts either a new chapter or a new part depending on the class loaded. To decide what version should be prepared, we test the \chapter command on existence.

```
\@ifundefined{chapter}{%
```

The case of an article-like class. Zero-level section is the \part.

```
  \def\NCC@startmainsec#1#2#3#4#5{%
    \NCC@preparesectag{\leavevmode#2}{\partname\nobreakspace}%
    \NCC@secskip{#3}%
    \secdef{\NCC@part{#1}{#4}{#5}}{\NCC@spart{#1}{#4}{#5}}%
  }
```

\NCC@spart Prepare the starred version of part:

\NCC@spart{⟨*alignment*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}{⟨*heading*⟩}

```
  \def\NCC@spart#1#2#3#4{%
    \let\@svsec\@empty
    \NCC@makepart{#1}{#3}{#4}{#2}{}%
    \NCC@sec@reset@controls
  }
```

\NCC@part Prepare the non-starred version of part:

\NCC@part{⟨*alignment*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}[⟨*toc-entry*⟩]{⟨*heading*⟩}

```
\def\NCC@part#1#2#3[#4]#5{%
  \def\NCC@make{\NCC@makepart{#1}{#3}{#5}{#2}}%
  \NCC@makesection{part}{\z@}{#4}{%
    \addcontentsline{toc}{part}{%
      \ifnum \c@secnumdepth>\m@ne \numberline{\thepart}\fi
      #4%
    }%
  }%
}
```

\NCC@makepart This command makes a part.

\NCC@makepart{⟨*alignment*⟩}{⟨*style*⟩}{⟨*heading*⟩}{⟨*afterskip*⟩}{⟨*action*⟩}

The \@svsec is either \@empty or contains a part tag.

```
\def\NCC@makepart#1#2#3#4#5{%
  \begingroup \normalfont
    \NCC@asecnumset
    \NCC@makeparttitle{#1}{#2}{#3}%
  \endgroup
  #5%
  \par\nobreak \vskip #4\relax \@afterheading \ignorespaces
}
```

\NCC@makeparttitle This command makes a part title itself. The \NCC@secmain hook contains the dynamic alignment style or nothing.

\NCC@makeparttitle{⟨*alignment*⟩}{⟨*style*⟩}{⟨*heading*⟩}

```
\def\NCC@makeparttitle#1#2#3{%
  \ifx\@svsec\@empty \else
    \NCC@secmain#1{\let\NCC@asecnum\@empty\@svsec\@@par}\nobreak
  \fi
  \interlinepenalty \@M \NCC@secmain#1{#2{#3\@@par}}%
}
```

\NCC@partsection Define the \NCC@partsection to be equal to the \NCC@mainsection command which will be specified later when a main section will be declared.

```
\def\NCC@partsection{\NCC@mainsection}
}{
```

The case of a book-like class. Zero-level section is the \chapter.

```
\def\NCC@startmainsec#1#2#3#4#5{%
  \NCC@startchap
  \NCC@preparesectag{\leavevmode#2}{\@chapapp\nobreakspace}%
  \secdef{\NCC@chapter{#1}{#3}{#4}{#5}}{\NCC@schapter{#1}{#3}{#4}{#5}}%
}
```

`\NCC@startchap` The start chapter hook:

```
\def\NCC@startchap{%
  \if@openright\cleardoublepage\else\clearpage\fi
  \thispagestyle{plain}\global\@topnum\z@
}
```

`\NCC@schapter` Prepare the starred version of chapter:

> `\NCC@schapter{`⟨*alignment*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`

```
\def\NCC@schapter#1#2#3#4#5{%
  \let\@svsec\@empty
  \NCC@makechapter{#1}{#2}{#4}{#5}{#3}{}%
  \NCC@sec@reset@controls
}
```

`\NCC@chapter` Prepare the non-starred version of chapter:

> `\NCC@chapter{`⟨*alignment*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`
> `[`⟨*toc-entry*⟩`]{`⟨*heading*⟩`}`

It uses the `\NCC@infloats{`⟨*action*⟩`}` hook that applies the specified action for all registered float types.

```
\def\NCC@chapter#1#2#3#4[#5]#6{%
  \@ifundefined{if@mainmatter}{}{\if@mainmatter\else\noheadingtag\fi}%
  \def\NCC@make{\NCC@makechapter{#1}{#2}{#4}{#6}{#3}}%
  \NCC@makesection{chapter}{\z@}{#5}{%
    \typeout{\@chapapp\space\thechapter.}%
    \addcontentsline{toc}{chapter}{%
      \ifnum \c@secnumdepth>\m@ne
        \numberline{\NCC@thetocchapter}\fi
      #5%
    }%
    \NCC@infloats{\addtocontents{\@nameuse{ext@\@captype}}%
                 {\protect\runinsectionskip}}%
  }%
}
```

`\beforechapter` The `\beforechapter{`⟨*something*⟩`}` hook is applied to the nearest chapter. An empty value of its parameter means no hook.

```
\newcommand\beforechapter[1]{\gdef\NCC@beforechapter{#1}}
\beforechapter{}
```

`\NCC@thetocchapter` The following hook allows redefine the appearance of chapter name in the TOC:

```
\def\NCC@thetocchapter{\thechapter}
```

`\NCC@makechapter` This command makes a chapter:

> `\NCC@makechapter{`⟨*alignment*⟩`}{`⟨*beforeskip*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`
> `{`⟨*afterskip*⟩`}{`⟨*action*⟩`}`

The `\@svsec` is either `\@empty` or contains a chapter tag.

```
\def\NCC@makechapter#1#2#3#4#5#6{%
  \if@twocolumn
    \@topnewpage[\NCC@makechaphead{#1}{#2}{#3}{#4}{#5}]%
  \else
    \NCC@makechaphead{#1}{#2}{#3}{#4}{#5}%
  \fi
  \NCC@makechapfinal{#6}%
  \@afterheading
  \ignorespaces
}
```

`\NCC@makechapfinal`
`\NCC@makechapfinalgobble`

The `\NCC@makechapfinal` hook applies a final action which can contain the `\chaptermark` command. Its default value is to put the parameter as is. If you let this command to be equal to the `\NCC@makechapfinalgobble`, the chapter mark will contain no chapter name.

```
\let\NCC@makechapfinal\@firstofone
\def\NCC@makechapfinalgobble#1{%
  \let\NCC@makechapmark\NCC@makemark
  \def\NCC@makemark{%
    \let\NCC@temp\@chapapp
    \let\@chapapp\@gobble
    \NCC@makechapmark
    \let\@chapapp\NCC@temp
  }%
  #1%
}
\@onlypreamble\NCC@makechapfinalgobble
```

`\NCC@makechaphead`

This command makes a chapter head:

`\NCC@makechaphead{`⟨*alignment*⟩`}{`⟨*beforeskip*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`
`{`⟨*afterskip*⟩`}`

```
\def\NCC@makechaphead#1#2#3#4#5{%
  \ifx\NCC@beforechapter\@empty
    \StartFromTextArea \vskip #2%
  \else
    \begingroup
      \@twocolumnfalse
      \let\NCC@global\@empty
      \NCC@sec@reset@controls
      \normalfont \NCC@beforechapter \par
    \endgroup
    \beforechapter{}%
  \fi
  \begingroup \normalfont
    \NCC@asecnumset
    \NCC@makechaptitle{#1}{#3}{#4}%
  \endgroup
  \par\nobreak \vskip #5\relax
}
```

`\NCC@makechaptitle` This command makes a chapter title itself:

```
\NCC@makechaptitle{⟨alignment⟩}{⟨style⟩}{⟨heading⟩}
```

The `\NCC@secmain` hook contains the dynamic alignment style or nothing.

```
\def\NCC@makechaptitle#1#2#3{%
  \ifx\@svsec\@empty \else
    \NCC@secmain#1{\let\NCC@asecnum\@empty\@svsec\@@par}%
  \fi
  \interlinepenalty \@M \NCC@secmain#1{#2{#3\@@par}}%
}
```

`\epigraph` `\epigraphparameters`

```
\epigraph[⟨width⟩]{⟨text⟩}{⟨author⟩}
\epigraphparameters{⟨style⟩}{⟨width⟩}{⟨height⟩}{⟨author-style⟩}
                   {⟨after-action⟩}
```

```
\newcommand*\epigraph[1][\NCC@epigraphwidth]{\NCC@epigraph{#1}}
\newcommand*\epigraphparameters[5]{%
  \def\NCC@epigraphwidth{#2}%
  \long\def\NCC@epigraph##1##2##3{
    \beforechapter{\def\epigraphwidth{##1}%
      #1\par
      \NCC@makeepigraph{#3}{##2}{#4{##3}}\par
      #5%
    }%
  }%
}
```

`\NCC@makeepigraph`

```
\NCC@makeepigraph{⟨height⟩}{⟨text⟩}{⟨author⟩}
```

```
\long\def\NCC@makeepigraph#1#2#3{%
  \@begin@tempboxa\vtop{\setlength{\hsize}{\epigraphwidth}%
      \@parboxrestore{#2\@@par}#3\@@par
    }%
    \setlength\@tempdima{#1}\advance\@tempdima -\totalheight
    \ifdim\@tempdima>\z@
      \advance\@tempdima\depth
      \dp\@tempboxa\@tempdima
    \fi
    \leavevmode\box\@tempboxa
  \@end@tempboxa
}
```

9.6 Make Part in Book-like Classes

`\NCC@startpart` The start-part hook:

```
\def\NCC@startpart{%
  \if@openright\cleardoublepage\else\clearpage\fi
  \thispagestyle{plain}%
}
```

`\NCC@spart` Prepare the starred version of part:

`\NCC@spart{`⟨*before*⟩`}{`⟨*after*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`

```
\def\NCC@spart#1#2#3#4{%
  \let\@svsec\@empty
  \NCC@makepart{#1}{#3}{#4}{#2}{}%
  \NCC@sec@reset@controls
}
```

`\NCC@part` Prepare the non-starred version of part:

`\NCC@part{`⟨*before*⟩`}{`⟨*after*⟩`}{`⟨*style*⟩`}[`⟨*toc-entry*⟩`]{`⟨*heading*⟩`}`

```
\def\NCC@part#1#2#3[#4]#5{%
  \def\NCC@make{\NCC@makepart{#1}{#3}{#5}{#2}}%
  \NCC@makesection{part}{\m@ne}{#4}{%
    \addcontentsline{toc}{part}{%
      \ifnum \c@secnumdepth>-2
        \numberline{\NCC@thetocpart}\fi
      #4%
    }%
  }%
}
```

`\NCC@thetocpart` The following hook allows redefine the appearance of part name in the TOC:

```
\def\NCC@thetocpart{\thepart}
```

`\NCC@makepart` This command makes a part:

`\NCC@makepart{`⟨*before*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}{`⟨*after*⟩`}{`⟨*action*⟩`}`

The `\@svsec` is either `\@empty` or contains a part tag.

```
\def\NCC@makepart#1#2#3#4#5{%
  \if@twocolumn \onecolumn \@tempswatrue \else \@tempswafalse \fi
  \begingroup\normalfont
    \NCC@asecnum@
    \NCC@makeparttitle{#1}{#2}{#3}%
  \endgroup
  \NCC@makepartfinal{#5}#4%
  \if@tempswa \twocolumn \fi
}
```

`\NCC@makepartfinal` `\NCC@makepartfinalgobble` The `\NCC@makepartfinal` hook applies a final action which can contain the `\partmark` command. Its default value is to put the parameter as is. If you let this command to be equal to the `\NCC@makepartfinalgobble`, the chapter mark will contain no chapter name.

```
\let\NCC@makepartfinal\@firstofone
```

```
  \def\NCC@makepartfinalgobble#1{%
    \let\NCC@makepartmark\NCC@makemark
    \def\NCC@makemark{%
      \let\NCC@temp\partname
      \let\partname\@gobble
      \NCC@makepartmark
      \let\partname\NCC@temp
    }%
    #1%
  }
  \@onlypreamble\NCC@makepartfinalgobble
```

`\NCC@makeparttitle` This command makes a part title itself:

`\NCC@makeparttitle{`⟨*before*⟩`}{`⟨*style*⟩`}{`⟨*heading*⟩`}`

```
  \def\NCC@makeparttitle#1#2#3{#1%
    \ifx\@svsec\@empty \else
      {\let\NCC@asecnum\@empty\@svsec\@@par}\nobreak \fi
    \interlinepenalty \@M #2{#3\@@par}
  }
}
```

9.7 Make Captions

`\CaptionTagSuffix` `\CaptionTagSuffix{`⟨*suffix*⟩`}` sets a default suffix after caption tag:

```
\newcommand*{\CaptionTagSuffix}[1]{\def\NCC@acapnum{#1}}
\@onlypreamble\CaptionTagSuffix
```

`\captiontagsuffix` `\captiontagsuffix[`⟨*type*⟩`]{`⟨*suffix*⟩`}` changes a suffix after caption tag.

```
\newcommand*\captiontagsuffix[2][]{%
  \NCC@prepare@capkey{suffix}{#1}{\def\NCC@acapnum{#2}}%
}
\let\NCC@capsuffix@\@empty
```

`\captionstyle` `\captionstyle[`⟨*type*⟩`]{`⟨*style*⟩`}` selects a style to be applied to the caption text. Three styles are available now: `default`, `center`, and `centerlast`.

```
\newcommand*\captionstyle[1][]{%
  \NCC@set@capkey{style}{style}{#1}%
}
```

`\captiontagstyle` `\captiontagstyle[`⟨*type*⟩`]{`⟨*style*⟩`}` selects a style of caption tag: `right` or `para`.

```
\newcommand*\captiontagstyle[1][]{%
  \NCC@set@capkey{tag}{tag style}{#1}%
}
```

`\captionwidth` The `\captionwidth[`⟨*type*⟩`]{`⟨*length*⟩`}` specifies a caption width in `\@tempdima`. Default width is `\linewidth`.

```
\newcommand*\captionwidth[2][]{%
  \NCC@prepare@capkey{width}{#1}{\setlength{\@tempdima}{#2}}%
}
```

\NCC@set@capkey `\NCC@set@capkey{`⟨*key*⟩`}{`⟨*description*⟩`}{`⟨*type*⟩`}{`⟨*value*⟩`}`

```
\def\NCC@set@capkey#1#2#3#4{%
  \@ifundefined{NCC@makecap#1@#4}
  {\PackageError{nccsect}{Unknown caption #2 `#4'}{}%
  }{%
    \edef\@tempa{\noexpand\NCC@prepare@capkey{#1}{#3}{%
        \expandafter\noexpand\csname NCC@makecap#1@#4\endcsname
      }%
    }%
    \@tempa
  }%
}
```

\NCC@prepare@capkey `\NCC@prepare@capkey{`⟨*key*⟩`}{`⟨*type*⟩`}{`⟨*definition*⟩`}`

```
\def\NCC@prepare@capkey#1#2{%
  \def\@tempa{#2}%
  \ifx\@tempa\@empty
    \ifx\@captype\@undefined \else \let\@tempa\@captype \fi
  \fi
  \expandafter\def\csname NCC@cap#1@\@tempa\endcsname
}
```

\NCC@apply@cap `\NCC@apply@cap{`⟨*key*⟩`}`

```
\def\NCC@apply@cap#1{%
  \@ifundefined{NCC@cap#1@\@captype}%
    {\let\@tempa\@empty}{\let\@tempa\@captype}%
  \csname NCC@cap#1@\@tempa\endcsname
}
```

\NCC@startcaption This command starts a caption:

`\NCC@startcaption{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`

```
\def\NCC@startcaption#1#2#3{%
  \secdef{\NCC@caption{#1}{#2}{#3}}{\NCC@scaption{#1}{#2}{#3}}%
}
```

\NCC@scaption Starred version of caption:

`\NCC@scaption{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}{`⟨*text*⟩`}`

```
\long\def\NCC@scaption#1#2#3#4{%
  \let\@svsec\@empty
  \NCC@makecaption{#3}{#1}{#4}{#2}{}%
  \NCC@sec@reset@controls
}
```

\NCC@caption Non-starred version of caption:

`\NCC@caption{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}[`⟨*toc-entry*⟩`]{`⟨*text*⟩`}`

```
\long\def\NCC@caption#1#2#3[#4]#5{%
  \def\NCC@make{\NCC@makecaption{#3}{#1}{#5}{#2}}%
  \NCC@makesection{\@captype}{\z@}{#4}{%
    \begingroup
      \let\centering\@empty
      \addcontentsline{\@nameuse{ext@\@captype}}{\@captype}{%
        \ifnum \c@secnumdepth>\m@ne
          \numberline{\@nameuse{the\@captype}}\fi
        #4%
      }%
    \endgroup
  }%
}
```

`\NCC@makecaption` This command makes a caption:

`\NCC@makecaption{`⟨*style*⟩`}{`⟨*beforeskip*⟩`}{`⟨*text*⟩`}{`⟨*afterskip*⟩`}{`⟨*action*⟩`}`

The `\@svsec` is either `\@empty` or contains a caption tag.

```
\long\def\NCC@makecaption#1#2#3#4#5{%
  \begingroup\par\normalfont
    #1{}\addvspace{#2}\noindent
```

Calculate in `\@tempcnta` caption variants: 0 – no caption, 1 – caption tag only, 2 – caption text only, 3 – both caption tag and text are nonempty.

```
    \ifx\@svsec\@empty \@tempcnta\z@ \else \@tempcnta\@ne \fi
    \def\@tempa{#3}%
    \ifx\@tempa\@empty \else \advance\@tempcnta\tw@ \fi
```

Put caption in a parbox aligned at the top line.

```
    \ifnum\@tempcnta=\z@ \else
      \NCC@apply@cap{suffix}%
      \NCC@apply@cap{width}%
      \NCC@vtopcap{\@parboxrestore\NCC@apply@cap{tag}{#3}\@@par}\par
```

We avoid insert zero skip after parbox to allow captions of side-by-side figures to be aligned at their top line.

```
      \setlength\@tempskipa{#4}%
      \ifdim\@tempskipa=\z@ \else \vskip \@tempskipa\fi
    \fi
  \endgroup
  #5%
}
```

`\NCC@vtopcap` `\NCC@vtopcap{`⟨*text*⟩`}` places a text in a vertical top-aligned box. Its width is prepared in `\@tempdima` before this macro. If its width is greater than the `\linewidth`, we allow overlap the box out of line. The overlapping directions are calculated from stretches of paragraph marginal skips.

```
\def\NCC@vtopcap#1{%
  \ifdim\@tempdima>\linewidth
    \@tempskipa \leftskip \advance\@tempskipa -1\@tempskipa
    \@tempskipb \rightskip \advance\@tempskipb \parfillskip
    \advance\@tempskipb -1\@tempskipb
    \vtop{\hb@xt@\linewidth{%
      \NCC@ifzeroskip\@tempskipa{}{\hss}%
      \vtop{\hsize\@tempdima#1}%
      \NCC@ifzeroskip\@tempskipb{}{\hss}%
    }}%
  \else
    \vtop{\hsize\@tempdima#1}%
  \fi
}
```

`\NCC@ifzeroskip` `\NCC@ifzeroskip{`⟨*register*⟩`}{`⟨*true-clause*⟩`}{`⟨*false-clause*⟩`}` executes the ⟨*true-clause*⟩ if the value of skip register is exactly zero skip without stretchability. Otherwise, the ⟨*false-clause*⟩ is executed.

```
\def\NCC@ifzeroskip#1{%
  \edef\@tempa{\the#1}\edef\@tempb{\the\z@skip}%
  \ifx\@tempa\@tempb
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
}
```

`\NCC@makecaptag@para` The `\NCC@makecaptag@para{`⟨*text*⟩`}` prepares run-in tag.

```
\long\def\NCC@makecaptag@para#1{%
  \ifnum\@tempcnta<\thr@@ \let\NCC@acapnum\@empty\fi
  \NCC@apply@cap{style}{{\@svsec}\ignorespaces#1}%
}
```

`\NCC@makecaptag@left` The `\NCC@makecaptag@left{`⟨*text*⟩`}` prepares flush-left tag.

```
\def\NCC@makecaptag@left{\NCC@separate@captag\raggedright}
```

`\NCC@makecaptag@center` The `\NCC@makecaptag@center{`⟨*text*⟩`}` prepares centered tag.

```
\def\NCC@makecaptag@center{\NCC@separate@captag\centering}
```

`\NCC@makecaptag@right` The `\NCC@makecaptag@right{`⟨*text*⟩`}` prepares flush-right tag.

```
\def\NCC@makecaptag@right{\NCC@separate@captag\raggedleft}
```

`\NCC@separate@captag` The `\NCC@separate@captag{`⟨*style*⟩`}{`⟨*text*⟩`}` prepares a caption tag in a separate line.

```
\long\def\NCC@separate@captag#1#2{%
  \ifodd\@tempcnta
    {\let\NCC@acapnum\@empty #1\@svsec\@@par}%
  \fi
  \ifnum\@tempcnta>\@ne
    \ifnum\@tempcnta=\thr@@ \vskip .5ex\fi
    \NCC@apply@cap{style}{#2}%
  \fi
}
```

`\NCC@makecapstyle@default` The `\NCC@makecapstyle@default{`⟨*text*⟩`}` prepares caption text in default LaTeX's style.

```
\long\def\NCC@makecapstyle@default#1{%
  \setbox\@tempboxa\vtop{\hsize\linewidth\@parboxrestore#1\@@par}%
  \ifdim\dp\@tempboxa<\baselineskip \centering#1%
  \else \box\@tempboxa \fi
}
```

`\NCC@makecapstyle@para` The `\NCC@makecapstyle@para{`⟨*text*⟩`}` prepares ordinary caption.

```
\long\def\NCC@makecapstyle@para#1{#1}
```

`\NCC@makecapstyle@left` The `\NCC@makecapstyle@left{`⟨*text*⟩`}` prepares flush-left caption.

```
\long\def\NCC@makecapstyle@left#1{\raggedright#1}
```

`\NCC@makecapstyle@right` The `\NCC@makecapstyle@right{`⟨*text*⟩`}` prepares flush-right caption.

```
\long\def\NCC@makecapstyle@right#1{\raggedleft#1}
```

`\NCC@makecapstyle@center` The `\NCC@makecapstyle@center{`⟨*text*⟩`}` prepares centered caption.

```
\long\def\NCC@makecapstyle@center#1{\centering#1}
```

`\NCC@makecapstyle@centerlast` The `\NCC@makecapstyle@centerlast{`⟨*text*⟩`}` prepares caption with last line centered.

```
\long\def\NCC@makecapstyle@centerlast#1{%
  \leftskip\@flushglue \rightskip -\@flushglue
  \parfillskip\z@\@plus 2fil\relax#1%
}
```

`\RegisterFloatType` The `\RegisterFloatType{`⟨*type*⟩`}` command registers a float type:

```
\newcommand*{\RegisterFloatType}[1]{%
  \edef\NCC@floatlist{\NCC@floatlist{#1}}%
}
\let\NCC@floatlist\@empty
\@onlypreamble\RegisterFloatType
```

`\NCC@infloats` The `\NCC@infloats{`⟨*action*⟩`}` command applies the given ⟨*action*⟩ to all registered float types. During the cycle, the `\@captype` contains a name of float and the `\@tempcnta` is equal to its registration number (1 for the figure float, 2 for the table float, and so on).

```
\def\NCC@infloats#1{%
  \@tempcnta\z@
  \let\NCC@temp \@captype
  \expandafter \@tfor \expandafter \@captype
    \expandafter :\expandafter =\NCC@floatlist \do
  {\advance\@tempcnta\@ne #1}%
  \let\@captype\NCC@temp
}
```

9.8 Declare Sections and Captions

`\DeclareSection` Now we can implement the `\DeclareSection` command. It generates:

`\NCC@mainsection` command if ⟨*level*⟩ = 0;

`\NCC@section`⟨*level-in-roman*⟩ command if ⟨*level*⟩ > 0;

`\NCC@cap@`⟨*float-type*⟩ command if ⟨*level*⟩ < 0.

```
\newcommand{\DeclareSection}{\@ifstar{\NCC@dsecx}{\NCC@dsec}}
\def\NCC@dsec#1#2{%
  \@ifnextchar[{\NCC@dsect{#1}{#2}}{\NCC@dsect{#1}{#2}[\z@skip]}%
}
\@onlypreamble\DeclareSection
\@onlypreamble\NCC@dsec
```

`\NCC@dsect` The non-starred version of section declaration command prepares display sections with traditional formatting:

`\NCC@dsect{`⟨*level*⟩`}{`⟨*type*⟩`}[`⟨*indent*⟩`]{`⟨*prefix*⟩`}{`⟨*beforeskip*⟩`}`
`{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`

It is also used for generation of run-in sections and captions.

```
\def\NCC@dsect#1#2[#3]#4#5#6#7{%
  \ifnum#1>\z@
    \expandafter\def\csname NCC@section\romannumeral#1\endcsname{%
      \NCC@setsectionsuffix{#1}%
      \NCC@preparesectag{#4}{}%
      \let\NCC@makesec\NCC@makesect
      \NCC@startsection{#2}{#1}{#3}{#5}{#6}{#7}}%
  \else
    \ifnum#1=\z@
      \def\NCC@mainsection{%
        \NCC@setsectionsuffix\z@
        \let\NCC@secmain\@empty
```

The empty `\NCC@secmain` means standard alignment of main section

```
        \NCC@startmainsec{%
          \NCC@hangfrom{\hskip #3}\NCC@adjsecmargins{}\@flushglue
          \ignorespaces}%
          {#4}{#5}{#6}{#7}%
      }%
    \else
      \NCC@dsecf{#2}{#4}{#5}{#6}{#7}%
    \fi
  \fi
}
\@onlypreamble\NCC@dsect
```

`\NCC@dsecx` The starred version of section declaration command prepares display sections with dynamic formatting:

\NCC@dsecx{⟨*level*⟩}{⟨*type*⟩}{⟨*prefix*⟩}{⟨*beforeskip*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}

It can also be used for generation of captions.

```
\def\NCC@dsecx#1#2#3#4#5#6{%
  \ifnum#1>\z@
    \expandafter\def\csname NCC@section\romannumeral#1\endcsname{%
      \NCC@setsectionsuffix{#1}%
      \NCC@setsectionstyle{#1}%
      \NCC@preparesectag{#3}{}%
      \let\NCC@makesec\NCC@makesecx
      \NCC@startsection{#2}{#1}{\z@}{#4}{#5}{#6}}%
  \else
    \ifnum#1=\z@
```

The non-empty \NCC@secmain hook means the dynamic alignment. We also redefine the dynamic section style \NCC@sec in such a way that the right skip stretchability will be 1fil if the section style has no flush glue.

```
      \def\NCC@mainsection{%
        \NCC@setsectionsuffix\z@
        \NCC@setsectionstyle\z@
        \let\NCC@secsave\NCC@sec \let\NCC@sec\NCC@secflush
        \def\NCC@secmain{\protect\NCC@sec{}}%
        \NCC@startmainsec{}{#3}{#4}{#5}{#6}%
      }%
    \else
      \NCC@dsecf{#2}{#3}{#4}{#5}{#6}%
    \fi
  \fi
}
\@onlypreamble\NCC@dsecx
```

\NCC@secflush

\NCC@secflush{⟨*tag*⟩} applies a section style saved in the \NCC@secsave macro and adjusts \rightskip and \parfillskip if left and right margins have no stretchability in sum. To adjust the right skip, we do the same tricks as in \NCC@adjsecmargins.

```
\def\NCC@secflush#1{\NCC@secsave{#1}%
  \@tempskipa\leftskip \advance\@tempskipa\rightskip
  \advance\@tempskipa -1\@tempskipa
  \NCC@ifzeroskip\@tempskipa{%
    \@tempskipa 1\rightskip \advance\@tempskipa -\rightskip
    \advance\@tempskipa \@flushglue
    \advance\rightskip \@tempskipa
    \advance\parfillskip -\@tempskipa
  }{}%
  \ignorespaces
}
```

\NCC@dsecf

This command declares captions of floats:

\NCC@dsecf{⟨*type*⟩}{⟨*prefix*⟩}{⟨*beforeskip*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}

```
\def\NCC@dsecf#1#2#3#4#5{%
  \expandafter\def\csname NCC@cap@#1\endcsname{%
    \def\NCC@makesectag####1{#2{\csname #1name\endcsname}%
      \nobreakspace####1\NCC@acapnum}%
    \NCC@startcaption{#3}{#4}{#5}%
  }%
}
\@onlypreamble\NCC@dsectf
```

`\DeclarePart` In book-like classes, a part is declared in a special way:

`\DeclarePart{`⟨*before*⟩`}{`⟨*after*⟩`}{`⟨*prefix*⟩`}{`⟨*style*⟩`}`

Long parameters are allowed in this declaration.

```
\@ifundefined{chapter}{}{%
  \newcommand\DeclarePart[4]{%
    \def\NCC@partsection{%
      \NCC@startpart
      \NCC@preparesectag{\leavevmode#3}{\partname\nobreakspace}%
      \secdef{\NCC@part{#1}{#2}{#4}}{\NCC@spart{#1}{#2}{#4}}%
    }%
  }
  \@onlypreamble\DeclarePart
}
```

9.9 Caption Patches

`\@makecaption` We emulate here the `\@makecaption{\fnum@`⟨*type*⟩`}{`⟨*caption*⟩`}` command to provide the compatibility with packages using it. It calls the `\NCC@cap@`⟨*type*⟩ command using the type specified in `\@captype` command. The counter is already stepped before this command and all necessary things are written to aux. Therefore, we turn off writing to aux and decrease a value of the float counter by `-1` because it will be stepped within again.

```
\long\def\@makecaption#1#2{%
  \begingroup
    \skipwritingtoaux
    \addtocounter\@captype\m@ne
    \csname NCC@cap@\@captype\endcsname[]{#2}%
  \endgroup
}
```

Add patch to the **supertabular** package:

```
\AfterPackage{supertabular}{%
  \long\def\ST@caption#1[#2]#3{\par%
    \addcontentsline{\csname ext@#1\endcsname}{#1}%
      {\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
    \begingroup\centering
      \def\@captype{#1}%
      \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
    \endgroup
  }
}
```

Add patch to the **xtab** package:

```
\AfterPackage{xtab}{%
  \long\def\ST@caption#1[#2]#3{\par%
    \@initisotab
    \addcontentsline{\csname ext@#1\endcsname}{#1}%
      {\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
    \begingroup\centering
      \def\@captype{#1}%
      \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
    \endgroup
    \global\advance\ST@pageleft -\PWSTcapht
    \ST@trace\tw@{Added caption. Space left for xtabular: \the\ST@pageleft}%
  }
}
```

Add patch to the **longtable** package:

```
\AfterPackage{longtable}{%
  \def\LT@makecaption#1#2#3{%
    \LT@mcol\LT@cols c{\hbox to\z@{\hss
      \parbox[t]\LTcapwidth{\centering\def\@captype{table}%
        \ifx#1\@gobble \NCC@cap@table*{#3}%
        \else \@makecaption{\fnum@table}{#3}%
        \fi
      }%
      \hss
    }}%
  }
}
```

9.10 Declare TOC-Entries

\DeclareTOCEntry The toc-entries declaration command:

`\DeclareTOCEntry{`⟨*level*⟩`}{`⟨*action*⟩`}{`⟨*prefix*⟩`}{`⟨*prototype*⟩`}`
`{`⟨*style*⟩`}[`⟨*next*⟩`]`

```
\newcommand*{\DeclareTOCEntry}[5]{%
  \@ifnextchar[{\NCC@dtoc{#1}{#2}{#3}{#4}{#5}}%
               {\NCC@dtoc{#1}{#2}{#3}{#4}{#5}[\NCC@nexttocnum{#3}{#4}]}%
}
\def\NCC@dtoc#1#2#3#4#5[#6]{%
```

Declare a toc-entry command for a registered float. We scan the registration list and find the necessary float type comparing its registration number with the negation of level. The generated command is `\l@`⟨*float-type*⟩:

```
  \ifnum#1<\z@
    \@tempswatrue
    \NCC@infloats{%
      \ifnum#1=-\@tempcnta
        \expandafter\def\csname l@\@captype\endcsname
          {\NCC@tocentry\z@{#2}{#3}{#4}{#5}}%
        \@tempswafalse
        \@break@tfor
      \fi
    }%
```

Incorrect level number. Generate an error.

```
    \if@tempswa
      \@tempcnta#1\relax
      \@tempcnta -\@tempcnta
      \PackageError{nccsect}%
        {Float type registration number \the\@tempcnta\space
         is out of range}{}%
    \fi
  \else
```

Prepare in `\@tempa` a command name: `\l@section` or `\l@subsection` or ... or `\l@subparagraph` or `\l@section@vi` or ...:

```
    \ifnum#1>\z@
      \edef\@tempa{\noexpand\def\expandafter\noexpand
                   \csname l@\NCC@secname{#1}\endcsname}%
```

or `\l@part` or `\l@chapter`:

```
    \else
      \@ifundefined{chapter}{\def\@tempa{\def\l@part}}%
                            {\def\@tempa{\def\l@chapter}}%
    \fi
```

Declare the toc-entry:

```
    \@tempa{\NCC@tocentry{#1}{#2}{#3}{#4}{#5}}%
```

Prepare in the `\l@tocskip@`⟨*next-level-in-roman*⟩ command the left margin adjustment command. The `\NCC@tocnumprototype{`⟨*style*⟩`}{`⟨*prototype*⟩`}` hook applies a style to the prototype of toc-entry number.

```
    \@tempcnta #1\relax \advance\@tempcnta\@ne
    \expandafter\def\csname l@tocskip@\romannumeral\@tempcnta
      \endcsname{\NCC@tocadj{\NCC@tocnumprototype{#5}{#6}}}%
  \fi
}
\@onlypreamble\DeclareTOCEntry
\@onlypreamble\NCC@dtoc
\def\NCC@nexttocnum#1#2{#1#2\NCC@atocnum}
\def\NCC@tocnumprototype#1#2{\let\applystyle\@firstoftwo#1{#2}}
```

`\NCC@tocadj` The command increases `\@tempdimb` on the width of the argument:

```
\def\NCC@tocadj#1{%
  \settowidth\@tempdima{\let\NCC@atocdo\@firstoftwo#1}%
  \advance\@tempdimb\@tempdima
}
```

`\DeclareTOCPart` Part toc-entry declaration in book-like classes. If optional ⟨*afterskip*⟩ is omitted, the default `\NCC@runskip` value is applied after this entry.

`\DeclareTOCPart{`⟨*action*⟩`}[`⟨*afterskip*⟩`]{`⟨*prefix*⟩`}{`⟨*prototype*⟩`}{`⟨*style*⟩`}`

```
\@ifundefined{chapter}{}{%
  \newcommand*\DeclareTOCPart[1]{%
    \@ifnextchar[{\NCC@dtocpart{#1}}{\NCC@dtocpart{#1}[\NCC@runskip]}%
  }
  \def\NCC@dtocpart#1[#2]#3#4#5{%
    \def\l@part##1##2{%
      \NCC@tocentry{-1}{%
```

We temporary add `1fil` to the toc right margin to prepare a ragged right toc-entry.

```
        \TOCMarginDrift{1fil}%
```

Breaks before part are preferred.

```
        \addpenalty{-\@highpenalty}#1%
      }{#3}{#4}{#5}{##1}{\hss##2}%
      \nobreak \vskip #2\relax
      \@nobreaktrue
      \everypar{\@nobreakfalse\everypar{}}%
    }%
  }
  \@onlypreamble\DeclareTOCPart
  \@onlypreamble\NCC@dtocpart
}
```

`\NCC@tocentry` This command makes a toc-entry:

`\NCC@tocentry{`⟨*level*⟩`}{`⟨*action*⟩`}{`⟨*prefix*⟩`}{`⟨*prototype*⟩`}`
`{`⟨*style*⟩`}{`⟨*entry*⟩`}{`⟨*page-number*⟩`}`

```
\def\NCC@tocentry#1#2#3#4#5#6#7{%
  \ifnum #1>\c@tocdepth \else
    \par\begingroup\normalfont #2%
      \let\applystyle\@firstoftwo
```

Calculate the left margin in the `\@tempdimb` register applying the `\l@tocskip@i`, . . . , `\l@tocskip@`⟨*level-in-roman*⟩ commands:

```
      \@tempdimb\z@ \@tempcnta #1\relax
      \@whilenum \@tempcnta >\z@\do
        {\@nameuse{l@tocskip@\romannumeral \@tempcnta}%
         \advance\@tempcnta\m@ne}%
```

The `\NCC@preparetocnum{`⟨*style*⟩`}{`⟨*prefix*⟩`}` hook prepares the `\NCC@maketocnum{`⟨*tag*⟩`}` command creating a number-line tag:

```
      \NCC@preparetocnum{#5}{#3}%
```

Calculate the hang indent value in `\@tempdima`:

```
      \settowidth\@tempdima{\let\NCC@atocdo\@firstoftwo\NCC@maketocnum{#4}}%
```

Produce the toc-entry. The `\NCC@tocentrytitle{`⟨*style*⟩`}{`⟨*title*⟩`}` hook applies the style to the toc-entry title.

```
      \@dottedtocline{#1}{\@tempdimb}{\@tempdima}%
        {\let\NCC@atocdo\@secondoftwo\NCC@tocentrytitle{#5}{#6\unskip}}%
        {\let\applystyle\@secondoftwo#5{#7}}%
```

Allow break after toc-entry:

```
      \@nobreakfalse
    \endgroup
  \fi
}
\def\NCC@preparetocnum#1#2{%
  \def\NCC@maketocnum##1{\NCC@atocdo{#1}{}{#2##1\NCC@atocnum}}%
}
\def\NCC@tocentrytitle#1#2{#1{\ignorespaces#2}}
```

`\numberline` Redefine the `\numberline{`⟨*tag*⟩`}` command to work correct if the width of tag is greater than `\@tempdima`. The tag is prepared with the `\NCC@maketocnum{`⟨*tag*⟩`}` command.

```
\DeclareRobustCommand*\numberline[1]{%
  \setbox\@tempboxa\hbox{\NCC@maketocnum{#1}}%
  \ifdim \wd\@tempboxa > \@tempdima
    \box\@tempboxa
  \else
    \hb@xt@\@tempdima{\unhbox\@tempboxa\hfil}%
  \fi
  \ignorespaces
}
```

`\NCC@maketocnum` The default implementation of the `\NCC@maketocnum{`⟨*tag*⟩`}` command. We must define it because the `\numberline` command must work out of scope of toc-entries.

```
\def\NCC@maketocnum#1{#1\NCC@atocnum}
\let\NCC@atocdo\@secondoftwo
```

`\NumberlineSuffix` The `\NumberlineSuffix{`⟨*calc-suffix*⟩`}{`⟨*actual-suffix*⟩`}` command saves suffices inserted after number tag in the `\numberline` command. It saves it in the `\NCC@atocnum` hook as parameters of `\NCC@atocdo` command. Letting the last one to `\@firstoftwo` or `\@secondoftwo`, we select the ⟨*calc-suffix*⟩ or ⟨*actual-suffix*⟩ respectively.

```
\newcommand*{\NumberlineSuffix}[2]{\def\NCC@atocnum{\NCC@atocdo{#1}{#2}}}
\@onlypreamble\NumberlineSuffix
```

`\TOCMarginDrift` The `\TOCMarginDrift{`⟨*drift*⟩`}` specifies allowed drift of right margin in TOC.

```
\newcommand*\TOCMarginDrift[1]{%
  \def\@tempa{#1}%
  \ifx\@tempa\@empty \let\NCC@tocdrift\@empty
  \else \def\NCC@tocdrift{\@plus #1\relax}\fi
}
```

\PnumPrototype The \PnumPrototype{⟨*prototype*⟩} command saves the page number prototype in the \NCC@pnum hook and applies the \NCC@setpnum command.

```
\newcommand*{\PnumPrototype}[1]{\def\NCC@pnum{#1}\NCC@setpnum}
\@onlypreamble\PnumPrototype
\def\NCC@setpnum{%
  \settowidth\@tempdima{\NCC@pnum}%
  \edef\@pnumwidth{\the\@tempdima}%
  \advance\@tempdima 1em
  \edef\@tocrmarg{\the\@tempdima \noexpand\NCC@tocdrift}%
}
```

\SetTOCStyle The toc-style hook is embedded into the \@starttoc command. We also recalculate the page number prototype and update margins when a toc starts.

```
\newcommand*{\SetTOCStyle}[1]{\def\NCC@tocstyle{#1}}
\@onlypreamble\SetTOCStyle
\let\NCC@latexstarttoc\@starttoc
\def\@starttoc#1{%
  \begingroup
    \normalfont \NCC@tocstyle \NCC@setpnum
    \NCC@latexstarttoc{#1}%
  \endgroup
}
```

9.11 Service and Defaults

\StartFromTextArea \StartFromHeaderArea These commands are applied at the beginning of page to set current position exactly at the first line of text area or at the header line, respectively. Both these commands are defined in two packages: in this one and in the **textarea**. To be sure that the commands are specified in these packages only, we mutually test packages to be loaded.

```
\@ifpackageloaded{textarea}{}{%
  \newcommand\StartFromTextArea{\par
    {\parskip\z@ \strut\par}\vskip -\baselineskip
  }
  \newcommand\StartFromHeaderArea{%
    \StartFromTextArea
    \vskip -\headsep \vskip -\ht\strutbox
  }
}
```

\bff The \bff command tries to set everything bold.

```
\newcommand{\bff}{\normalfont\bfseries\mathversion{bold}}
```

\aftersectionvspace This command eliminates a vertical space inserted after a previous section and inserts a vertical space specified.

```
\newcommand*\aftersectionvspace[1]{%
  \ifvmode \if@nobreak
    \vskip -\lastskip \vskip #1\relax
  \fi \fi
}
```

\startsection Define the `\startsection` command. In article-class, both zero and negative levels refer to the same part section.

```
\newcommand*{\startsection}[1]{%
  \ifnum#1>\z@
    \def\@tempa{\csname NCC@section\romannumeral#1\endcsname}%
  \else
    \ifnum#1=\z@
      \def\@tempa{\NCC@mainsection}%
    \else
      \def\@tempa{\NCC@partsection}%
    \fi
  \fi
  \@tempa
}
```

\part \section \subsection \subsubsection \paragraph \subparagraph Set aliases for almost all section levels, except chapter. The part is called here as a section of a negative level.

```
\def\part{\startsection\m@ne}
\def\section{\startsection\@ne}
\def\subsection{\startsection\tw@}
\def\subsubsection{\startsection\thr@@}
\def\paragraph{\startsection4}
\def\subparagraph{\startsection5}
```

\caption Redefine the `\caption` command. We do this at the beginning of document to reject possible redefinitions of captions in other packages such as `float`. I think this is not the `float`'s responsibility to decide where a caption must go on: before or after the float body. And what about complicated floats consisting of side floats and etc.? We also reset to zero the `\abovecaptionskip` and `\belowcaptionskip` registers if they are specified to provide partial compatibility with the `float` package. If the registers are not specified (as in `ncc` class), they are emulated with macros.

```
\AtBeginDocument{%
  \def\caption{%
    \ifx\@captype\@undefined
      \@latex@error{\noexpand\caption outside float}\@ehd
      \expandafter\@gobble
    \else
      \expandafter\@firstofone
    \fi
    {\csname NCC@cap@\@captype\endcsname}%
  }%
  \@ifundefined{abovecaptionskip}{\def\abovecaptionskip{\z@}}%
               {\abovecaptionskip\z@}%
  \@ifundefined{belowcaptionskip}{\def\belowcaptionskip{\z@}}%
               {\belowcaptionskip\z@}%
}
```

Registration of standard floats:

```
\RegisterFloatType{figure}
\RegisterFloatType{table}
```

Declare all sections and captions except the part and chapter:

```
\DeclareSection{-2}{table}{}{\z@}{10pt}{}
\DeclareSection{-1}{figure}{}{10pt}{\z@}{}
\DeclareSection*1{section}{}%
               {3.5ex \@plus 1ex \@minus .2ex}%
               {2.3ex \@plus .2ex}{\Large\bff}
\DeclareSection*2{subsection}{}%
               {3.25ex \@plus 1ex \@minus .2ex}%
               {1.5ex \@plus .2ex}{\large\bff}
\DeclareSection*3{subsubsection}{}%
               {3ex \@plus 1ex \@minus .2ex}%
               {1.5ex \@plus .2ex}{\normalsize\bff}
\DeclareSection4{paragraph}{}%
               {\NCC@runskip}{-1em}{\normalsize\bff}
\DeclareSection5{subparagraph}[\parindent]{}%
               {\NCC@runskip}{-1em}{\normalsize\bff}
\@ifundefined{chapter}{
```

Declare the part and toc-entries for the article-like style:

```
  \DeclareSection*0{part}{\Large\bff}%
                 {5ex \@plus 1ex \@minus .2ex}%
                 {4ex \@plus .2ex}{\huge\bff}
  \DeclareTOCEntry{-2}{}{}{9}{}% table
  \DeclareTOCEntry{-1}{}{}{9}{}% figure
  \DeclareTOCEntry0{\runinsectionskip\def\@dotsep{1000}}{}{III}{\bff}[]
  \DeclareTOCEntry1{\runinsectionskip}{}{9}{}
  \DeclareTOCEntry2{}{}{9.9}{}
  \DeclareTOCEntry3{}{}{9.9.9}{}
}{
```

`\ChapterPrefixStyle` Specify the appearance of chapter prefix in the toc and the header.

```
  \newcommand*{\ChapterPrefixStyle}[1]{%
    \def\NCC@thetocchapter{\thechapter}%
    \let\NCC@makechapfinal\NCC@makechapfinalgobble
    \@for\@tempa:=#1\do{%
      \@ifundefined{NCC@chapin@\@tempa}{%
        \PackageError{nccsect}{Unknown style `\@tempa'\MessageBreak
        Only the `toc' and `header' styles are allowed}{}%
      }{\csname NCC@chapin@\@tempa\endcsname}%
    }%
  }
  \def\NCC@chapin@toc{\def\NCC@thetocchapter{\@chapapp\ \thechapter}}
  \def\NCC@chapin@header{\let\NCC@makechapfinal\@firstofone}
  \@onlypreamble\ChapterPrefixStyle
  \@onlypreamble\NCC@chapin@toc
  \@onlypreamble\NCC@chapin@header
```

`\chapter` Declare the part, the chapter, toc-entries for the book-like style, and specify default epigraph parameters:

```
  \def\chapter{\startsection\z@}
  \DeclarePart{\StartFromTextArea\vfil\centering}%
              {\vfil\newpage \if@twoside\if@openright
                 \mbox{}\thispagestyle{empty}\newpage\fi\fi}%
              {\vspace{4ex}\huge\bff}{\Huge\bff}
  \DeclareSection*0{chapter}{\vspace{3ex}\huge\bff}{10ex}%
                  {8ex \@plus .2ex}{\Huge\bff}
  \DeclareTOCEntry{-2}{}{}{9.9}{}% table
  \DeclareTOCEntry{-1}{}{}{9.9}{}% figure
  \DeclareTOCPart{\NCC@secskip{4ex \@plus .2ex}\def\@dotsep{1000}}
                 {}{II}{\large\bff}
  \DeclareTOCEntry0{\runinsectionskip\def\@dotsep{1000}%
                     \aftergroup\penalty\aftergroup\@highpenalty}{}{9}{\bff}
  \DeclareTOCEntry1{}{}{9.9}{}[9.9]
  \DeclareTOCEntry2{}{}{9.9.9}{}[9.9.9]
  \DeclareTOCEntry3{}{}{}{}[\qquad]
  \epigraphparameters{\StartFromHeaderArea\small\raggedleft}%
                      {.45\linewidth}{5\baselineskip}%
                      {\raggedleft\itshape}{\vspace{2ex}}
}
```

Declare other toc-entries:

```
\DeclareTOCEntry4{}{}{}{}[\qquad]
\DeclareTOCEntry5{}{}{}{}[\qquad]
```

Set defaults:

```
\noindentaftersection
\sectionstyle{hangindent}
\SectionTagSuffix{\quad}
\RunningSectionSuffix{}
\captionwidth{\linewidth}
\captionstyle{default}
\captiontagstyle{para}
\CaptionTagSuffix{:\hskip .7em \@plus .2em \@minus .1em}
\NumberlineSuffix{\quad}{\enskip}
\PnumPrototype{99}
\TOCMarginDrift{}
\SetTOCStyle{}
⟨/package⟩
```