The nccsect package*†

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2006/01/19

Contents

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.

*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.

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.

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;

[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.

`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 particular 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 on 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}
```

Analogously to the `centerlast` style, the `rightlast` style (last line is adjusted to the right) can be easy created:

```
\newplainsectionstyle{rightlast}{0pt}[r]{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` mod-

ifier 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}
```