indextra — Enhanced index typesetting*

Alan J. Cain†

Released 2025-10-26

Abstract

This package provides some enhanced features for typesetting indexes, notably: (1) Continuation text when entries or sub-entries continue from one page or column to the next. (2) An interface for accessing marks created from index entries, so that (for example) a running head can include the range of index entries that appears on the page.

Contents

*This file describes v0.21.6, last revised 2025-10-26.

†Email: a.j.cain (AT) gmail.com

1 Introduction

By default, an index in a LaTeX document is typeset in two columns and provides no indication if a page or column break occurs inside an index entry or sub-entry. The package repeatindex[1] handles a page or column break that occurs between sub-entries within the same entry by repeating the keyword of the main entry. But it does not repeat the keyword of a sub-entry when a page or column break occurs within it. Nor does it repeat the keyword of an entry (at the top level) if the break occurs mid-way through a long list of pages or ranges for that entry directly. In *The TeXbook*, Knuth described a plain TeX method to create continuation texts using the mark mechanism [1, pp. 261–3], which does not have these limitations. But there are still situations in which this method can cause problems: for instance, if the keyword in the index entry spans two or more lines, it may break across a page or column and the continuation text would appear in the middle of the keyword.

The indextra package is intended to create suitable continuation texts for all of these possibilities. An example is shown in Figure 1.1, where the list of locations within a sub-entry is too long to fit in the left-hand column. At the top of the right-hand column, indextra has indicated that both the main entry and the sub-entry have both

[1] URL: https://ctan.org/pkg/repeatindex

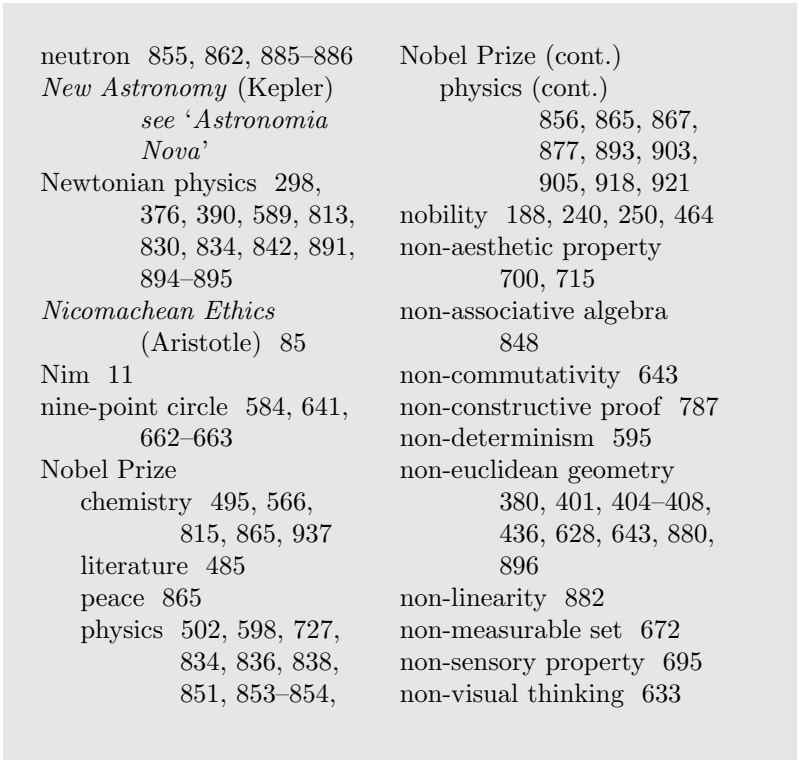

Figure 1.1: An example of how indextra continues an entry and a sub-entry after a column or page break.

been continued. If there had been a page break as well as a column break, the same indicators would have been placed at the start of the left-hand column of the next page.

indextra tries to break columns intelligently. If only one line of a multi-line index entry would fit in the remaining space of the current column, the column break is inserted first, since there would be no saving in space because the continutation text would occupy (at least) one line of the next column. Similarly, it will not break a column before or during any cross-references in the index entry.

indextra also supplies a system for retrieving index entry keywords as marks, for use in running heads to show the range of index entries on the current page. The keywords of the first and last top-level entries on each page are available as `\indextrafirstmark` and `\indextralastmark`.

Caveat: Although indextra has been used successfully for the indexes of the author's books,[2] in its current state it should be regarded as semi-experimental, requiring further development, and having many limitations and some incompatibilites (see Subsection 9.2).

[2] Available on the Internet Archive under Creative Commons licences:
(1) A.J. Cain, *Form & Number: A History of Mathematical Beauty*. Lisbon, 2024. URL: https://archive.org/details/cain_formandnumber_ebook_large
(2) G.H. Hardy, *An Annotated Mathematician's Apology*. With annotations and commentary by A.J. Cain. Lisbon, 2019. URL: https://archive.org/details/hardy_annotated

2 Requirements

indextra does not depend on any other packages, but requires a recent LaTeX kernel with expl3 support. (Any kernel version since 2020-02-02 should suffice.)

3 Installation

To install indextra manually, run `tex indextra.ins` and copy the file `indextra.sty` to somewhere LaTeX can find it, `indextra.ist` to somewhere *MakeIndex* and/or `upmendex` can find it, and `indextra.xdy` to somewhere `xindy` can find it. To build the documentation, first compile `indextra-doc-demo.tex` to a PDF, then compile `indextra-doc.tex`.

4 Getting started

4.1 LaTeX

On the LaTeX side, simply use the package with `\usepackage{indextra}`. (There are no package options.) Use `\makeindex` and `\index` as as usual. Note that indextra is incompatible with imakeindex.

If the hyperref package is used, set `hyperindex=false` in its package options or in `\hypersetup`, because indextra has its own mechanism to add index hyperlinks, activated by its own `hyperindex` key.

4.2 Index generation

The `.ind` file (generated by *MakeIndex*, `upmendex`, or `xindy`) must be specially formatted for indextra. The style file `indextra.ist` is supplied for use with *MakeIndex* and `upmendex`. It can be used with:

```
makeindex -s indextra.ist ⟨filename⟩.idx
```

or

```
upmendex -s indextra.ist ⟨filename⟩.idx
```

Also supplied is a barebones `xindy` module `indextra.xdy` style file, which should be used with other modules to set up sorting and attributes. The module `indextra.xdy` should be specified last when invoking `xindy`, to override any previous specification of how to write to the `.idx` file. For example, one might use:

```
xindy -M makeidx.xdy -M utf8.xdy -M indextra.xdy ⟨filename⟩.idx
```

5 Configuration

`\indextrasetup` `\indextrasetup{⟨options⟩}`

This command is used to specify options. The argument ⟨*options*⟩ is a key–value list. The options are described in the subsections below and are summarized in Table 5.1.

Table 5.1: Summary of keys that can be set using `\indextrasetup`.

Key name	Value	Default
`before code`	LaTeX code	`\begin{theindex}`
`after code`	LaTeX code	`\end{theindex}`
`level 0 style`	LaTeX code	`\parindent=0em\hangindent=3.75em`
`level 1 style`	LaTeX code	`\parindent=1.5em\hangindent=5.25em`
`level 2 style`	LaTeX code	`\parindent=3em\hangindent=6.75em`
`separator keyword crossref`	LaTeX code	`\break`
`separator keyword location`	LaTeX code	`\space\space`
`separator crossref crossref`	LaTeX code	`\break`
`separator crossref location`	LaTeX code	`\break`
`separator location location`	LaTeX code	`,␣`
`crossref macros`	List of macros	`\see\seealso`
`hyperindex`	{`true`, `false`}	`true`
`bookmarks`	{`true`, `false`}	`true`

5.1 Before and after code

`before code` This option specifies code that is executed at the start of the index (more specifically, at the start of the `theindextra` environment in the generated `.ind` file). The code will be executed before indextra sets various parameters and makes available the commands described in Section 8. The user may wish to use this option to set up running heads using indextra's marks (see Subsection 9.3 for an example). (*Default:* `\begin{theindex}`)

`after code` This option specifies code that is executed at end start of the index (more specifically, at the end of the `theindextra` environment in the generated `.ind` file). (*Default:* `\begin{theindex}`)

5.2 Styles

`level` n `style` The style applied to an index entry at level $n = 0, 1, 2$ (that is, respectively to an entry, a sub-entry, and a sub-sub-entry).
(*Default:* `level 0 style=\parindent=0em\hangindent=3.75em`;
`level 1 style=\parindent=1.5em\hangindent=5.25em`;
`level 2 style=\parindent=3em\hangindent=6.75em`)

5.3 Separators

indextra always typesets cross-references before locations in each index entry, sub-entry, or sub-sub-entry. The following options specify the separators to be used.

`separator keyword crossref` The separator to be placed between the keyword of an index entry and a cross-reference. (*Default:* `\break`)

`separator keyword location` The separator to be placed between the keyword of an index entry and a location (that is, a page or range of pages), when the first location immediately follows the keyword (when there are no cross-references). (*Default:* `\space\space`)

`separator crossref crossref` The separator to be placed between the two cross-references in an index entry. (*Default:* `\break`)

`separator crossref location` The separator to be placed between the last cross-reference in an entry and the first location. (*Default:* `\break`)

`separator location` The separator to be placed between two locations in an index entry. (*Default:* `,␣`)
`location`

5.4 Specification of cross-references

`crossref macros` A list of macros that signify cross-references. (*Default:* `\see\seealso`)

5.5 Hyperlinks

`hyperindex` A boolean `true`/`false` that indicates whether index locations should be hyperlinked if the `\hyperpage` macro from the hyperref package is available. Note that a user redefinition of `\indextrapage` or `\indextrarange` (see Subsection 7.2) will override this setting. If hyperlinks are required in this case, the user's redefinition should create them. (*Default:* `true`)

5.6 Bookmarks

`bookmarks` A boolean `true`/`false` that indicates whether the beginning of each group of index entries (usually terms beginning with a particular letter) should be bookmarked if the `\belowpdfbookmark` macro from the hyperref package is available. (*Default:* `true`)

5.7 Headings

`headings` A boolean `true`/`false` that indicates whether there should be a heading at the beginning of each group of index entries (usually terms beginning with a particular letter). This heading will be created by the user-redefinable macro `\indextramakegroupheading` (see Subsection 7.5). (*Default:* `true`)

6 Marks

indextra uses the keyword of each (top-level) index entry to insert a mark. The following macros make available the first and last marks on the current page. They are intended for use in a running header to show the range of entries on the current page.

`\indextrafirstmark` The first mark on the current page, with the macro `\indextramakemark` applied.

`\indextralastmark` The last mark on the current page, with the macro `\indextramakemark` applied.

7 User-redefinable macros

7.1 Keywords

The following macros is used to typeset keywords of entries, sub-entries, and sub-sub-entries. It can be redefined by the user.

\indextrakeyword `\indextrakeyword{⟨level⟩}{⟨keyword⟩}`

This macro is used to typeset a keyword for an entry of level ⟨*level*⟩. The default definition simply yields ⟨*keyword*⟩. Users may wish to use a redefinition to apply styling or for other purposes. The result will already have the code specified in `level` ⟨*level*⟩ `style` applied.

7.2 Page and range typesetting

The following two macros are actually used to typeset locations. They can be redefined by the user.

\indextrapage `\indextrapage{⟨encapsulation⟩}{⟨page⟩}`

This command is used to typeset a reference to a single page. The default definition is effectively ⟨*encapsulation*⟩{⟨*page*⟩}, but using the configuration option `hyperindex=true` will create a hyperlink to the actual page. Any redefinition of this command will override the effect of `hyperindex=true`, so if the user still wishes the reference to be a hyperlink to the actual page, the new definition must create the hyperlink.

\indextrarange `\indextrarange{⟨encapsulation⟩}{⟨start⟩}{⟨end⟩}`

This command is used to typeset a reference to a range of pages. The default definition is effectively ⟨*encapsulation*⟩{⟨*page*⟩}--⟨*encapsulation*⟩{⟨*page*⟩}, but using the configuration option `hyperindex=true` will create a hyperlink to the actual pages. Users may wish to redefine `\indextrarange` to abbreviate ranges (so that, for instance, 1024–1025 is replaced by 1024–5). Any redefinition of this command will override the effect of `hyperindex=true`, so if the user still wishes the references to be hyperlinks to the actual pages, the new definition must create the hyperlinks.

7.3 Continuation text

indextra inserts continuation text for each entry, sub-entry, and sub-sub-entry that has been interrupted by a page or column break. The following macro creates the continuation text and can be redefined by the user.

\indextramakecontinuation `\indextramakecontinuation{⟨level⟩}{⟨keyword⟩}`

This macro is used to generate a continuation text when the entry at ⟨*level*⟩ with the supplied ⟨*keyword*⟩ contains a column break. The default definition simply yields ⟨*keyword*⟩ `(cont.)`. Users may wish to use a redefinition to apply styling. The continuation text will already have the code specified in `level` ⟨*level*⟩ `style` applied.

7.4 Marks

\indextramakemark `\indextramakemark{⟨text⟩}`

This command is applied to any mark retrieved via either `\indextrafirstmark` or `\indextralastmark`. The default definition simply yields ⟨*text*⟩. A redefinition could be used to abbreviate or otherwise process ⟨*text*⟩.

7.5 Headings

If the option `headings=true` is set, the following macro generates the group heading and can be redefined by the user.

`\indextramakegroupheading` `\indextramakegroupheading{`⟨*group*⟩`}`

This macro is used to generate a heading for ⟨*group*⟩. The default definition yields `\textbf{`⟨*group*⟩`}`. Users may wish to use a redefinition to apply a different style. Note that even if this command is redefined to yield nothing, an extra vertical space will still be produced in the index where the heading would have been. To disable headings and avoid this extra space, set `headings=false` using `\indextrasetup`.

8 Macros used in the generated `.ind` file

For completeness, this section documents the commands and the enclosing environment used in the generated `.ind` file. None of these is intended to be used or redefined by the user.

`theindextra` (*env.*) Environment containing the generated index.

The commands below are only defined inside the `theindextra` environment.

`\indextrastyleversion` `\indextrastyleversion{`⟨*version*⟩`}`

Specify the indextra version of the style used to generate the `.ind` file. (This macro exists in case future updates change the required format of the `.ind` file.)

`\indextraprocessortype` `\indextraprocessortype{`⟨*type*⟩`}`

Specify the type of the processor used to generate the `.ind` file. (Different encapsulations of ranges, which must be handled differently, are generated by *MakeIndex*/`upmendex` and by `xindy`.) `{`⟨*type*⟩`}` is either `ist`, indicating that *MakeIndex* or `upmendex` was used, or `xdy`, indicating that `xindy` was used.

`\indextraentry` `\indextraentry{`⟨*level*⟩`}{`⟨*keyword*⟩`}{`⟨*crossrefs-and-locations*⟩`}`

This command typesets an index entry, sub-entry, or sub-sub-entry (if ⟨*level*⟩ is respectively `0`, `1`, or `2`) with the given ⟨*keyword*⟩. The parameter ⟨*crossrefs-and-locations*⟩ is a comma-separated list of cross-references (using macros such as `\see` or `\seealso` as specified in the `crossref macros` option) and locations (meaning pages or ranges).

`\indextraspace` `\indextraspace`

This command produces a space of one line in the index.

`\indextragroup` `\indextragroup{`⟨*letter*⟩`}`

This command begins the new group ⟨*letter*⟩. Depending on configuration, it may produce a heading and/or a bookmark.

9 Usage notes and caveats

9.1 Typesetting process

indextra works by tracking how much space is left in each column and intially typesetting each index entry into a buffer and measuring it. If there is enough space left in the column, the buffer contents are added to the output and the amount of space left is adjusted appropriately. Otherwise, enough of the buffer as will fit is split off and output, then a column break is called, the continuation text is added to the top of the remaining content in the buffer, and the process continues in a new column.

This process relies on the fact that an index is a highly structured text with a restricted format. Any 'exotic' index entries may break the process.

9.2 Limitations and incompatibilies

- When choosing where to break a column, indextra only considers the current entry at the current level. It does not, for example, automatically decide to insert a column break before a one-line top-level entry that contains a sub-entry, even though this would be a more desirable result.
- indextra cannot cope with `\verb` commands in index keywords. (But succeeding in generating such an `.ind` via the usual LaTeX indexing commands would be a challenge.)
- indextra is incompatible with imakeidx, which uses the multicols package to set the index instead of the native LaTeX two-column mode. In particular, LaTeX marks cannot be set from within the `multicols` environment.
- indextra should not be used with repeatindex, and in any case replaces its functionality.
- indextra cannot be used if the usual indexing system is heavily customized. For example, the l3doc class uses its own specialized implementation of indexing and so indextra cannot be used alongside it.

9.3 Example of using marks

One could incorporate marks into the index running heads using by appending suitable code to the default value (`\begin{theindex}`) of the `before code` key:

```
\indextrasetup{
  before code={%
    \begin{theindex}%
      \markboth
      {\MakeUppercase\indexname~%
        \noexpand\indextrafirstmark--\noexpand\indextralastmark}
      {\MakeUppercase\indexname~%
        \noexpand\indextrafirstmark--\noexpand\indextralastmark}%
  },
}
```

(The `\noexpand` macros ensure that `\indextrafirstmark` and `\indextralastmark` are expanded when the page is shipped out, not when `\markboth` is used.)

References

[1] D.E. Knuth. *The TEXbook*. Addison–Wesley, 2021. *Computers & Typesetting*, vol. A.

10 Implementation

```
⟨*package⟩
⟨@@=indextra⟩
```

10.1 Initial set-up

Package identification/version information.

```
\NeedsTeXFormat{LaTeX2e}[2020-02-02]
\ProvidesExplPackage{indextra}{2025-10-26}{0.21.6}
  {Enhanced index typesetting}
```

10.2 User configuration

Set up the key–value options and the variables in which the settings will be stored.

\l__indextra_before_code_tl
\l__indextra_after_code_tl

Token list keys that store code to be executed before and after the index is typeset.

```
\keys_define:nn { indextra }
{
  before~code .tl_set:N = \l__indextra_before_code_tl,
  after~code .tl_set:N = \l__indextra_after_code_tl,
  before~code .initial:n = {\begin{theindex}},
  after~code .initial:n = {\end{theindex}},
}
```

(*End of definition for* `\l__indextra_before_code_tl` *and* `\l__indextra_after_code_tl`.)

\l__indextra_level_0_style_tl
\l__indextra_level_1_style_tl
\l__indextra_level_2_style_tl

Token list keys that store the style to be applied to entries at each level.

```
\keys_define:nn { indextra }
{
  level~0~style .tl_set:c = {l__indextra_level_0_style_tl},
  level~1~style .tl_set:c = {l__indextra_level_1_style_tl},
  level~2~style .tl_set:c = {l__indextra_level_2_style_tl},
  level~0~style .initial:n = {\parindent=0em\hangindent=3.75em},
  level~1~style .initial:n = {\parindent=1.5em\hangindent=5.25em},
  level~2~style .initial:n = {\parindent=3em\hangindent=6.75em},
}
```

(*End of definition for* `\l__indextra_level_0_style_tl`, `\l__indextra_level_1_style_tl`, *and* `\l__indextra_level_2_style_tl`.)

\l__indextra_separator_kw_cr_tl
\l__indextra_separator_kw_loc_tl
\l__indextra_separator_cr_cr_tl
\l__indextra_separator_cr_loc_tl
\l__indextra_separator_loc_loc_tl

Token list keys to store separators. Macros are abbreviated as follows `kw`: keyword; `cr`: cross-reference; `loc`: location (page or range).

```
\keys_define:nn { indextra }
{
  separator~keyword~crossref .tl_set:N = \l__indextra_separator_kw_cr_tl,
  separator~keyword~location .tl_set:N = \l__indextra_separator_kw_loc_tl,
  separator~crossref~crossref .tl_set:N = \l__indextra_separator_cr_cr_tl,
  separator~crossref~location .tl_set:N = \l__indextra_separator_cr_loc_tl,
  separator~location~location .tl_set:N = \l__indextra_separator_loc_loc_tl,
```

```
  separator~keyword~crossref .initial:n = {\break},
  separator~keyword~location .initial:n = {\space\space},
  separator~crossref~crossref .initial:n = {\break},
  separator~crossref~location .initial:n = {\break},
  separator~location~location .initial:n = {,~},
}
```

(*End of definition for* `\l__indextra_separator_kw_cr_tl` *and others.*)

`\l__indextra_crossref_macros_tl` Token list key to store macros that should count as cross-references.

```
\keys_define:nn { indextra }
{
  crossref~macros .tl_set:N = \l__indextra_crossref_macros_tl,
  crossref~macros .initial:n = {\see\seealso},
}
```

(*End of definition for* `\l__indextra_crossref_macros_tl`.)

`\l__indextra_hyperindex_bool` Boolean indicating whether locations should be hyperlinked to pages, if `\hyperpage` is available and the user has not redefined `\indextrapage` and `\indextrarange`.

```
\keys_define:nn { indextra }
{
  hyperindex .bool_set:N = \l__indextra_hyperindex_bool,
  hyperindex .initial:n = {true},
}
```

(*End of definition for* `\l__indextra_hyperindex_bool`.)

`\l__indextra_bookmarks_bool` Boolean indicating whether groups should be bookmarked, if `\belowpdfbookmark` is available.

```
\keys_define:nn { indextra }
{
  bookmarks .bool_set:N = \l__indextra_bookmarks_bool,
  bookmarks .initial:n = {true},
}
```

(*End of definition for* `\l__indextra_bookmarks_bool`.)

`\l__indextra_headings_bool` Boolean indicating whether group headings should appear.

```
\keys_define:nn { indextra }
{
  headings .bool_set:N = \l__indextra_headings_bool,
  headings .initial:n = {true},
}
```

(*End of definition for* `\l__indextra_headings_bool`.)

`\indextrasetup` User command to set key–value configuration.

```
\NewDocumentCommand{\indextrasetup}{ m }
{
  \keys_set:nn{ indextra }{ #1 }
}
```

(*End of definition for* `\indextrasetup`. *This function is documented on page 4.*)

10.3 Environment and commands for the `.ind` file

Define the environment and commands used in the generated `.ind` file.

`theindextra` This environment is the analogue of the `theindex` environment defined by basic LaTeX.

```
\NewDocumentEnvironment{theindextra}{}
  { \__indextra_main_begin: }
  { \__indextra_main_end: }
%
```

(*End of definition for* `theindextra`*. This function is documented on page* ??.)

`\__indextra_main_begin:` This macro (locally) sets various TeX dimensions and defines the commands that are actually used in the index, namely `\indextrastyleversion`, `\indextraprocessortype`, `\indextraspace`, `\indextragroup`, and `\indextraentry`.

```
\cs_new:Npn \__indextra_main_begin:
{
```

Open a group and execute code for before the index has been typeset.

```
  \group_begin:
  \tl_use:N\l__indextra_before_code_tl
```

Since column breaks will be inserted manually, zero the penalties which influence column/page breaking.

```
  \int_zero:N\clubpenalty
  \int_zero:N\widowpenalty
```

Set `\splittopskip`, `\topskip`, and `\parskip` to the needed values, and set `\g__indextra_remaining_space_dim`, which tracks the space left in the current column, to the initial value of `\@colht`, which will be the height of the column after any initial two-column text has been typeset.

```
  \dim_gset:Nn\g__indextra_remaining_space_dim{\@colht}
  \dim_set:Nn\splittopskip{.7\baselineskip}
  \dim_set:Nn\topskip{.7\baselineskip}
  \dim_set:Nn\parskip{0pt}
```

Make available the appropriate macros.

```
  \cs_set_eq:NN\indextrastyleversion\__indextra_style_version:n
  \cs_set_eq:NN\indextraprocessortype\__indextra_processor_type:n
  \cs_set_eq:NN\indextraspace\__indextra_space:
  \cs_set_eq:NN\indextragroup\__indextra_group:n
  \cs_set_eq:NN\indextraentry\__indextra_entry:nnn
```

If user settings require it, or `\belowpdfbookmark` is unavailable, disable group bookmarks and/or headings.

```
  \bool_if:nF
    { \l__indextra_bookmarks_bool && \cs_if_exist_p:N\belowpdfbookmark }
    {
      \cs_set_eq:NN\__indextra_bookmark_stored_group:\prg_do_nothing:
    }
  \bool_if:NF\l__indextra_headings_bool
    {
      \cs_set_eq:NN\__indextra_typeset_stored_group_heading:\prg_do_nothing:
    }
```

Depending on user configuration, if `\hyperpage` is available and `\indextrapage` or `\indextrapage` have not been redefined, then change them to create hyperlinks.

```
\bool_if:nT{ \l__indextra_hyperindex_bool && \cs_if_exist_p:N\hyperpage }
  {
    \cs_if_eq:NNT\indextrapage\__indextra_page_basic:nn
      {
        \cs_set_eq:NN\indextrapage\__indextra_page_hyperref:nn
      }
    \cs_if_eq:NNT\indextrarange\__indextra_range_basic:nnn
      {
        \cs_set_eq:NN\indextrarange\__indextra_range_hyperref:nnn
      }
  }
}
```

(*End of definition for* `\__indextra_main_begin:`.)

`\__indextra_main_end:` Execute code for after the index has been typeset and close the group opened in `\__indextra_main_begin:`.

```
\cs_new:Npn \__indextra_main_end:
{
  \tl_use:N\l__indextra_after_code_tl
  \group_end:
}
```

(*End of definition for* `\__indextra_main_end:`.)

```
\msg_new:nnn{ indextra }{ incompatible_version }
  { The~.ind~file~was~generated~using~a~style~from~a~different~version~of~indextra. }
```

`\__indextra_style_version:n` Specify the version of indextra's index processor style file (`.ist` or `.xdy`) that was used to generate the `.ind` file. If is incompatible with version of indextra in use, an error will result.

```
\cs_new:Npn\__indextra_style_version:n #1
{
  \str_if_eq:nnF{#1}{0.1}
    {
      \msg_error:nn{ indextra }{ incompatible_version }
    }
}
```

(*End of definition for* `\__indextra_style_version:n`.)

`\__indextra_processor_type:n` Set the type of the processor used to generate the `.ind` file. *MakeIndex* and `upmendex` produce ranges of the form `\encapsulation{`⟨*start*⟩`--`⟨*end*⟩`}`, while `xindy` produces ranges of the form `\encapsulation{}--\encapsulation{}`. These kinds of ranges must be parsed differently, and while the kind of range could be detected when typesetting each location, there are efficiency savings from using a dedicated parser in each case. `#1` should be either `ist`, indicating that the processor was either *MakeIndex* or `upmendex`, or `xdy`, that the processor was `xindy`. This macro will be set equal to `\indextrasetprocessortype` inside the `theindextra` environment.

```
\cs_new:Npn\__indextra_processor_type:n #1
{
  \cs_set_eq:Nc\__indextra_typeset_aux_location:n
```

```
    { __indextra_typeset_#1_location:n }
}
```

(*End of definition for* `\__indextra_processor_type:n`.)

10.4 Column handling

`\g__indextra_remaining_space_dim` Dimension variable to hold the amount of space left in the current column.

```
\dim_new:N\g__indextra_remaining_space_dim
```

(*End of definition for* `\g__indextra_remaining_space_dim`.)

`\__indextra_new_column:` Fill any remaining space in the current column, start a new column, and set `\g__indextra_remaining_space_dim` to `\@colht` (which is the amount of space available in the new column).

```
\cs_new:Npn\__indextra_new_column:
  {
    \vfill
    \newpage
    \dim_gset:Nn\g__indextra_remaining_space_dim{\@colht}
  }
```

(*End of definition for* `\__indextra_new_column:`.)

10.5 Spaces

`\__indextra_space:` Create a space in the index, which will usually signify the end of one letter group and the beginning of another. This takes up space in the column, so the space (a `\strut`) is typeset via the same mechanism as index entries, namely `\__indextra_typeset_buffer:`.

```
\cs_new:Npn\__indextra_space:
  {
    \dim_compare:nNnT{\g__indextra_remaining_space_dim}<{\baselineskip}{
      \__indextra_new_column:
    }
    \vbox_set:Nn\l__indextra_buffer_box{\leavevmode\strut\par}
    \__indextra_typeset_buffer:
  }
```

(*End of definition for* `\__indextra_space:`.)

10.6 Groups

`\l__indextra_group_tl` Token-list variable to store the current group.

```
\tl_new:N\l__indextra_group_tl
```

(*End of definition for* `\l__indextra_group_tl`.)

`\__indextra_group:` Begin a letter group in the index. The macro simply stores the group title; the bookmark and/or heading will be created by the next index entry. This macro will be set equal to `\indextragroup` inside the `theindextra` environment.

```
\cs_new:Npn\__indextra_group:n #1
  {
    \tl_set:Nn\l__indextra_group_tl{#1}
  }
```

(*End of definition for* `\__indextra_group:`.)

`\g__indextra_group_bookmark_int` Integer variable to index groups; used in the creation of bookmarks in `\__indextra_bookmark_stored_group:`.

```
\int_new:N\g__indextra_group_bookmark_int
```

(*End of definition for* `\g__indextra_group_bookmark_int`.)

`\__indextra_bookmark_stored_group:` If `\l__indextra_group_tl` is non-empty, create a bookmark below the current level (i.e., using `\belowpdfbookmark`), using `\l__indextra_group_tl` for the bookmark text. The variable `\g__indextra_group_bookmark_int` is incremented at each call and used to create distinct bookmark names. (One cannot use the the content of `\l__indextra_group_tl` for the bookmark name, because (1) it might not be suitable as a name, and (2) there may be two indexes (e.g. names and subjects) which contain the same groups.)

```
\cs_new:Npn\__indextra_bookmark_stored_group:
  {
    \tl_if_empty:NF\l__indextra_group_tl
      {
        \int_gincr:N\g__indextra_group_bookmark_int
        \belowpdfbookmark
          {\tl_use:N\l__indextra_group_tl}
          {pdf:indextra:\int_value:w\g__indextra_group_bookmark_int}
      }
  }
```

(*End of definition for* `\__indextra_bookmark_stored_group:`.)

`\__indextra_typeset_stored_group_heading:` If `\l__indextra_group_tl` is non-empty, typeset its contents as a heading for the group.

```
\cs_new:Npn\__indextra_typeset_stored_group_heading:
  {
    \tl_if_empty:NF\l__indextra_group_tl
      {
        \group_begin:
        \noindent\strut
        \indextramakegroupheading{\tl_use:N\l__indextra_group_tl}
        \strut\par
        \group_end:
      }
  }
```

(*End of definition for* `\__indextra_typeset_stored_group_heading:`.)

`\indextramakegroupheading` Typeset a heading for the letter group passed as `#1`.

```
\cs_new:Npn\indextramakegroupheading #1
  {
    \textbf{#1}
  }
```

(*End of definition for* `\indextramakegroupheading`. *This function is documented on page 8.*)

10.7 Entries

\l__indextra_level_int

Integer variable to hold the level of the current entry.

```
\int_new:N\l__indextra_level_int
```

(*End of definition for* `\l__indextra_level_int`.)

\l__indextra_saved_keyword_0
\l__indextra_saved_keyword_1
\l__indextra_saved_keyword_2

Token list variables to save keywords at each level.

```
\tl_new:c{l__indextra_saved_keyword_0}
\tl_new:c{l__indextra_saved_keyword_1}
\tl_new:c{l__indextra_saved_keyword_2}
```

(*End of definition for* `\l__indextra_saved_keyword_0`, `\l__indextra_saved_keyword_1`, *and* `\l__indextra_saved_keyword_2`.)

\l__indextra_crossref_seq
\l__indextra_location_seq

Comma-separated list variables to hold the cross-references and locations of the current entry. These will be populated via a call to `\__indextra_filter_crossref_location_seqs:n`

```
\seq_new:N\l__indextra_crossref_seq
\seq_new:N\l__indextra_location_seq
```

(*End of definition for* `\l__indextra_crossref_seq` *and* `\l__indextra_location_seq`.)

__indextra_entry:nnn

Typeset an index entry. The three parameters are the level (`#1`), the keyword (`#2`), and the rest of the entry (`#3`), which should be a comma-separated list of cross-references and/or locations.

```
\cs_new:Npn\__indextra_entry:nnn #1#2#3
  {
```

First save the level and the keyword, and sort the rest of the entry into the cross-reference and location seqs.

```
    \int_set:Nn\l__indextra_level_int{#1}
    \tl_set:cn{l__indextra_saved_keyword_#1}{#2}
    \__indextra_filter_crossref_location_seqs:n{#3}
```

Typeset the entire entry (prefixed by any stored group heading) into `\l__indextra_buffer_box`. The bookmarking command is also executed here, since it is the contents of this box that will ultimately be shipped out.

```
    \vbox_set:Nn\l__indextra_buffer_box{
      \__indextra_bookmark_stored_group:
      \__indextra_typeset_stored_group_heading:
      \group_begin:
      \__indextra_set_style:n{#1}
      \leavevmode
      \strut
      \indextrakeyword{#1}{#2}
      \__indextra_typeset_between:
      \__indextra_typeset_locations:
      \strut
      \par
      \goodbreak
      \group_end:
    }
```

Now typeset some material into \l_tmpa_box and \l_tmpb_box, the dimensions of which will be used in computing the minimum acceptable portion of the entry to typeset into the current column. First, typeset the part of the entry where a column break is unacceptable (prefixed by any stored group heading) into \l_tmpa_box.

```
\vbox_set:Nn\l_tmpa_box{
  \__indextra_typeset_stored_group_heading:
  \group_begin:
  \__indextra_set_style:n{#1}
  \leavevmode
  \strut
  \indextrakeyword{#1}{#2}
  \__indextra_typeset_between:
  \__indextra_typeset_first_location:
  \strut
  \par
  \group_end:
}
```

Typeset any stored group heading into \l_tmpb_box.

```
\vbox_set:Nn\l_tmpb_box{
  \__indextra_typeset_stored_group_heading:
}
```

Measuring \l_tmpa_box, \l_tmpb_box, and \l__indextra_buffer_box, check whether there is enough space in the current column to typeset any stored heading and the minimum acceptable portion of the entry. Basically, a split should not occur within the part typeset into \l_tmpa_box, and if the entry as a whole is longer than one line, then at least two lines should be typeset (because if only one line can fit into the current column, there is probably no overall saving of space by putting it in the current column with a continuation in the next column). Note that if there is no stored heading, the height and depth of \l_tmpb_box are both 0 pt, so no conditional is needed.

```
\dim_compare:nNnT
  {\g__indextra_remaining_space_dim}
  <
  {
    \dim_min:nn{
      \dim_max:nn{
        \box_ht:N\l_tmpa_box
        +\box_dp:N\l_tmpa_box
      }{
        \box_ht:N\l_tmpb_box
        +\box_dp:N\l_tmpb_box
        +2\baselineskip
      }
    }{
      \box_ht:N\l__indextra_buffer_box
      +\box_dp:N\l__indextra_buffer_box
    }
  }
  {
```

Case: not enough space in the current column. Start a new column and, unless this is a top-level entry, insert the appropriate continuation into the \l__indextra_buffer_box.

```
    \__indextra_new_column:
```

```
        \int_compare:nNnT{#1}>{0}{
          \__indextra_vbox_prepend:Nn\l__indextra_buffer_box{
            \__indextra_make_continuation:n{\l__indextra_level_int-1}
          }
        }
      }
```

`\l__indextra_buffer_box` now contains the material that should be typeset onto the page, and there is enough space in the current column to typeset a minimal acceptable portion of it. Typesetting is handled by `\__indextra_set_buffer:`, which may use `\vsplit`, which produces a warning '`Underfull \vbox (badness 10000)`' with even a 1 sp mismatch between available breakpoints and desired height. For efficiency, `\__indextra_set_buffer:` uses an approximation to the desired height, so set `\vbadness` to 10000 to suppress these warnings.

```
    \group_begin:
    \int_set:Nn\vbadness{10000}
    \__indextra_typeset_buffer:
    \group_end:
```

Finally, clear any stored group.

```
    \tl_clear:N\l__indextra_group_tl
  }
```

(*End of definition for* `\__indextra_entry:nnn`.)

10.7.1 Filtering cross-references and locations

In each entry, cross-references and locations may be mixed up and need to be filtered into separate lists.

`\__indextra_filter_crossref_location_seqs:n`

Use the supplied list of cross-references and location (`#1`), which is a clist, to populate `\l__indextra_crossref_seq` and `\l__indextra_location_seq`. The list of macros that are the first tokens in cross-references is contained in `\l__indextra_crossref_macros_tl`. All that is required is to check whether the head token in each item in the `#1` is in `\l__indextra_crossref_macros_tl`. and assign that item to the correct seq.

```
\cs_new:Npn\__indextra_filter_crossref_location_seqs:n #1
  {
    \seq_clear:N\l__indextra_crossref_seq
    \seq_clear:N\l__indextra_location_seq

    \clist_map_inline:nn{#1}
      {
        \tl_if_in:NoTF
          \l__indextra_crossref_macros_tl{ \tl_head:w ##1 {} \q_stop }
          {
            \seq_put_right:Nn\l__indextra_crossref_seq{##1}
          }
          {
            \seq_put_right:Nn\l__indextra_location_seq{##1}
          }
      }
  }
```

(*End of definition for* `\__indextra_filter_crossref_location_seqs:n`.)

10.7.2 Style

__indextra_set_style:n Set up the style for an index entry at the given level.

```
\cs_new:Npn \__indextra_set_style:n #1
  {
    \tl_use:c{l__indextra_level_#1_style_tl}
  }
```

(*End of definition for* `\__indextra_set_style:n`.)

10.7.3 Typesetting keywords

\indextrakeyword Take a level (`#1`) and a keyword (`#2`) and return a (possibly styled) keyword.

```
\cs_new:Npn\indextrakeyword #1#2
  {
    #2
  }
```

(*End of definition for* `\indextrakeyword`. *This function is documented on page* 7.)

10.7.4 Typesetting cross-references

__indextra_typeset_between: Typeset everything between the keyword and the locations. If there are no cross-references, this means typesetting just the keyword–location separator if the list of locations is non-empty (it is possible that there are no locations; practically, this would happen when all the locations are in sub-entries); otherwise, it means typesetting the cross-references with the relevant separators before, after, and between.

```
\cs_new:Npn\__indextra_typeset_between:
  {
    \seq_if_empty:NTF\l__indextra_crossref_seq
      {
        \seq_if_empty:NF\l__indextra_location_seq
          { \tl_use:N\l__indextra_separator_kw_loc_tl }
      }
      {
        \tl_use:N\l__indextra_separator_kw_cr_tl
        \seq_use:Nn
          \l__indextra_crossref_seq
          { \tl_use:N\l__indextra_separator_cr_cr_tl }
        \seq_if_empty:NF\l__indextra_location_seq
          { \tl_use:N\l__indextra_separator_cr_loc_tl }
      }
  }
```

(*End of definition for* `\__indextra_typeset_between:`.)

10.7.5 Typesetting locations

\l__indextra_first_item_bool Boolean variable to track whether the current location is the first one in the entry, so that a separator can be inserted before every non-first location.

```
\bool_new:N\l__indextra_first_item_bool
```

(*End of definition for* `\l__indextra_first_item_bool`.)

__indextra_typeset_locations: Typeset locations stored in `\l__indextra_locations_seq`. Using `\seq_use:Nn` is not enough, because each entry must be processed separately. Thus separators must be inserted 'manually'.

```
\cs_new:Npn\__indextra_typeset_locations:
  {
    \bool_set_true:N\l__indextra_first_item_bool
    \seq_map_inline:Nn\l__indextra_location_seq
      {
        \bool_if:nF{\l__indextra_first_item_bool}
          { \tl_use:N\l__indextra_separator_loc_loc_tl }
        \__indextra_typeset_aux_location:n{##1}
        \bool_set_false:N\l__indextra_first_item_bool
      }
  }
```

(*End of definition for* `\__indextra_typeset_locations:`.)

__indextra_typeset_first␣location: Typeset the first location stored in `\l__indextra_locations_seq`.

```
\cs_new:Npn\__indextra_typeset_first_location:
  {
    \exp_args:Ne\__indextra_typeset_aux_location:n
      {\seq_item:Nn\l__indextra_location_seq{1}}
  }
```

(*End of definition for* `\__indextra_typeset_first location:`.)

__indextra_is_nonrange_p:w To be called in the form `\__indextra_is_nonrange_p:w`⟨*parameter*⟩`--\q_stop`; uses TEX parsing to check whether ⟨*parameter*⟩ does not contain a range marker `--`.

```
\cs_new:Npn\__indextra_is_nonrange_p:w #1--#2\q_stop
  {
    \tl_if_empty_p:n{#2}
  }
```

(*End of definition for* `\__indextra_is_nonrange_p:w`.)

s_new:Npn__indextra_typeset_ist_location:n Typeset a *MakeIndex*- or `upmendex`-style location. The location might be a

- a single page;
- an encapsulated single page `\encapsulation{`⟨*page*⟩`}`;
- a range; or
- an encapsulated range `\encapsulation{`⟨*start*⟩`--`⟨*end*⟩`}`.

```
\cs_new:Npn\__indextra_typeset_ist_location:n #1
  {
```

Check for an encapsulation (a leading control sequence, comparing catcode with `\prg_do_nothing:` simply as a convenience). Either call `\__indextra_typeset_ist_encap_page_or_range:Nn` with this encapsulation or with a 'do-nothing' encapsulation and the braced parameter.

```
    \tl_if_head_eq_catcode:nNTF{#1}\prg_do_nothing:
      {
        \__indextra_typeset_ist_encap_page_or_range:Nn #1
      }
```

```
      {
        \__indextra_typeset_ist_encap_page_or_range:Nn \prg_do_nothing:{#1}
      }
  }
```

(*End of definition for* `\cs_new:Npn\__indextra_typeset_ist_location:n`.)

`indextra_typeset_ist_encap_page_or_range:Nn` Take an encapsulation and a parameter and typeset the parameter as a page or range, as appropriate.

```
\cs_new:Npn\__indextra_typeset_ist_encap_page_or_range:Nn #1#2
  {
    \bool_if:nTF{\__indextra_is_nonrange_p:w #2--\q_stop}
      {
```

Case: not a range. Parameter `#2` a single page.

```
        \indextrapage{#1}{#2}
      }
      {
```

Case: range. Parameter `#2` is a range. Call `\__indextra_typeset_ist_encap_range:w` to parse and typeset it.

```
        \__indextra_typeset_ist_encap_range:w#1\q_mark #2\q_stop
      }
  }
```

(*End of definition for* `\__indextra_typeset_ist_encap_page_or_range:Nn`.)

`\__indextra_typeset_ist_encap_range:w` This is effectively just a helper macro so that TEX parsing can be used to convert the parameters into the form required by `\indextrarange`.

```
\cs_new:Npn\__indextra_typeset_ist_encap_range:w #1\q_mark #2--#3\q_stop
  {
    \indextrarange{#1}{#2}{#3}
  }
```

(*End of definition for* `\__indextra_typeset_ist_encap_range:w`.)

`s_new:Npn\__indextra_typeset_xdy_location:n` Typeset a `xindy`-style location. The location might be

- a single page;
- an encapsulated single page `\encapsulation{`⟨*page*⟩`}`;
- a range; or
- an encapsulated range `\encapsulation{`⟨*start*⟩`}--\encapsulation{`⟨*end*⟩`}`.

```
\cs_new:Npn\__indextra_typeset_xdy_location:n #1
  {
    \bool_if:nTF{\__indextra_is_nonrange_p:w #1--\q_stop}
      {
```

Case: not a range. Check for an encapsulation (a leading control sequence, comparing catcode with `\prg_do_nothing:` simply as a convenience). Either call `\__indextra_typeset_ist_encap_page_or_range:Nn` with this encapsulation or with a 'do-nothing' encapsulation and the braced parameter.

```
        \tl_if_head_eq_catcode:nNTF{#1}\prg_do_nothing:
          {
```

```
            \indextrapage #1
          }
          {
            \indextrapage \prg_do_nothing:{#1}
          }
      }
      {
```

Case: range.

```
        \__indextra_typeset_xdy_range:w #1\q_stop
      }
  }
```

(*End of definition for* `\cs_new:Npn\__indextra_typeset_xdy_location:n`.)

`\__indextra_typeset_xdy_range:w`

To be called in the form `\__indextra_typeset_xdy_range:w`⟨*parameter*⟩`\q_stop`, so that `#1` and `#2` will be the start and end of the range, possibly with encapsulations. Check `#1` for a leading control sequence. If it exists, assume the same holds for `#2`, and the parameters are of the form `\encapsulation{`⟨*start*⟩`}` and `\encapsulation{`⟨*end*⟩`}` and can be passed to `\__indextra_typeset_xdy_encap_range:w`. Otherwise, each parameter is a page, so call `\indextrarange` with 'do-nothing' encapsulations and braced parameters.

```
\cs_new:Npn\__indextra_typeset_xdy_range:w #1--#2\q_stop
  {
    \tl_if_head_eq_catcode:nNTF{#1}\prg_do_nothing:
      {
        \__indextra_typeset_xdy_encap_range:NnNn #1#2
      }
      {
        \indextrarange\prg_do_nothing:{#1}{#2}
      }
  }
```

(*End of definition for* `\__indextra_typeset_xdy_range:w`.)

`\__indextra_typeset_xdy_encap_range:NnNn`

This is effectively just a helper macro so that TEX parsing can be used to convert the parameters into the form required by `\indextrarange`.

```
\cs_new:Npn\__indextra_typeset_xdy_encap_range:NnNn #1#2#3#4
  {
    \indextrarange{#1}{#2}{#4}
  }
```

(*End of definition for* `\__indextra_typeset_xdy_encap_range:NnNn`.)

`\__indextra_page_basic:nn`
`\__indextra_range_basic:nnn`
`\__indextra_page_hyperref:nn`
`\__indextra_range_hyperref:nnn`

Default macros for typesetting a page and a range, depending on whether hyperref is loaded.

```
\cs_new:Npn\__indextra_page_basic:nn #1#2
  {
    #1{#2}
  }
\cs_new:Npn\__indextra_range_basic:nnn #1#2#3
  {
    #1{#2}--#1{#3}
  }
```

```
\cs_new:Npn\__indextra_page_hyperref:nn #1#2
  {
    #1{\hyperpage{#2}}
  }
\cs_new:Npn\__indextra_range_hyperref:nnn #1#2#3
  {
    #1{\hyperpage{#2}}--#1{\hyperpage{#3}}
  }
```

(*End of definition for* `\__indextra_page_basic:nn` *and others.*)

Set `\indextrapage` and `\indextrarange` to the 'basic' versions of the above macros. They may be set to the 'hyperref' versions at the start of the `theindextra` environment depending on user settings.

```
\cs_set_eq:NN\indextrapage\__indextra_page_basic:nn
\cs_set_eq:NN\indextrarange\__indextra_range_basic:nnn
```

10.8 Continuations

`\__indextra_make_continuation:n` Return a continuation text for the level specified in `#1`, indicating that this (and higher) levels are continued from the previous column. The continuation text is created using the stored keywords with `\indextramakecontinuation` applied to each.

```
\cs_new:Npn\__indextra_make_continuation:n #1
  {
    \int_step_inline:nnn{0}{#1}{
      \group_begin:
      \__indextra_set_style:n{##1}
      \leavevmode\strut
      \indextramakecontinuation{##1}{\tl_use:c{l__indextra_saved_keyword_##1}}
      \strut\par
      \group_end:
    }
  }
```

(*End of definition for* `\__indextra_make_continuation:n`.)

`\indextramakecontinuation` Take a level (`#1`) and a keyword (`#2`) and return a continuation text.

```
\cs_new:Npn\indextramakecontinuation #1#2
  {
    #2~(cont.)
  }
```

(*End of definition for* `\indextramakecontinuation`. *This function is documented on page* 7.)

10.9 Marks

Define a new mark class for use by indextra.

```
\mark_new_class:n{indextra}
```

`\__indextra_mark_insert:` Insert the top-level saved keyword as a mark.

```
\cs_new:Npn\__indextra_mark_insert:
  {
    \mark_insert:nn{indextra}{\tl_use:c{l__indextra_saved_keyword_0}}
  }
```

(*End of definition for* `\__indextra_mark_insert:`.)

`\indextramakemark` Macro to process an indextra mark for use. Any mark retrieved by `\indextrafirstmark` or `\indextralastmark` will have `\indextramakemark` applied.

```
\cs_new:Npn\indextramakemark #1
  {
    #1
  }
```

(*End of definition for* `\indextramakemark`. *This function is documented on page* 7.)

`\indextrafirstmark` `\indextralastmark` Retrieve the first and last indextra marks on the page, where 'first' and 'last' have their standard LaTeX meanings. The marks will be processed by `\indextramakemark`.

```
\cs_new:Npn\indextrafirstmark
  {
    \indextramakemark{ \mark_use_first:nn{page}{indextra} }
  }
\cs_new:Npn\indextralastmark
  {
    \indextramakemark{ \mark_use_last:nn{page}{indextra} }
  }
```

(*End of definition for* `\indextrafirstmark` *and* `\indextralastmark`. *These functions are documented on page* 6.)

10.10 Buffer

`\l__indextra_buffer_box` Box variable to hold remaining material to be typeset for the current index entry, sub-entry, or sub-sub-entry.

```
\box_new:N\l__indextra_buffer_box
```

(*End of definition for* `\l__indextra_buffer_box`.)

`\__indextra_typeset_buffer:` Output material that has been typeset into `\l__indextra_buffer_box` and update `\g__indextra_remaining_space_dim` as necessary. If necessary, split off as much material as will fit in the current column from `\l__indextra_buffer_box`, typeset it, add continuation text to the top of `\l__indextra_buffer_box`, then recursively call this macro. The recursion ends when all material in `\l__indextra_buffer_box` has been output.

```
\cs_new:Npn\__indextra_typeset_buffer:
  {
```

Insert a mark. This is either on the first call, or a recursive call, in which case the current position is at the top of the column. A mark should be inserted in any case.

```
    \__indextra_mark_insert:
```

Check whether it is necessary to split off some material for this column and start a new one. There is no need to check about the minimum acceptable split: if necessary, a new column has already been started in `\__indextra_entry:nnn`.

```
    \dim_compare:nNnT
    {\g__indextra_remaining_space_dim}
    <
    {\box_ht:N\l__indextra_buffer_box+\box_dp:N\l__indextra_buffer_box}
      {
```

Case: A split is necessary. Compute an approximation (definitely too large, but acceptably so) to the amount to be split off, then split and rebox.

```
      \dim_set:Nn
        \l_tmpa_dim{\g__indextra_remaining_space_dim-.3\baselineskip}
      \vbox_set_split_to_ht:NNn
        \l_tmpa_box\l__indextra_buffer_box{\l_tmpa_dim}
      \vbox_set:Nn
        \l_tmpa_box{\vbox_unpack_drop:N\l_tmpa_box}
```

Update `\g__indextra_remaining_space_dim` and output the split material.

```
      \dim_gset:Nn\g__indextra_remaining_space_dim
        {
          \g__indextra_remaining_space_dim
          -\box_ht:N\l_tmpa_box
          -\box_dp:N\l_tmpa_box
        }
      \vbox_unpack_drop:N\l_tmpa_box
```

Add a continuation text to the top of `\l__indextra_buffer_box`, start a new column, and recurse.

```
      \__indextra_vbox_prepend:Nn
        \l__indextra_buffer_box
        {\__indextra_make_continuation:n{\l__indextra_level_int}}
      \__indextra_new_column:
      \__indextra_typeset_buffer:
    }
    {
```

Case: No split is necessary. Update `\g__indextra_remaining_space_dim` and output the buffer contents.

```
      \dim_gset:Nn\g__indextra_remaining_space_dim
        {
          \g__indextra_remaining_space_dim
          -\box_ht:N\l__indextra_buffer_box
          -\box_dp:N\l__indextra_buffer_box
        }
      \vbox_unpack_drop:N\l__indextra_buffer_box
    }
  }
```

(*End of definition for* `\__indextra_typeset_buffer:`.)

`\__indextra_vbox_prepend` Takes a vertical box variable as `#1` and typeset the material in `#2` into it above the original content.

```
\cs_new:Npn\__indextra_vbox_prepend:Nn #1#2
{
  \vbox_set:Nn #1{
    \vbox{#2}
    \vbox_unpack_drop:N #1
  }
}
```

(*End of definition for* `\__indextra_vbox_prepend`.)

⟨/package⟩

10.11 Style files

10.11.1 `.ist` – *MakeIndex*, `upmendex`

⟨∗ist⟩

```
page_precedence "ArRn"

preamble         "\\begin{theindextra}\n  \\indextrastyleversion{0.1}\n  \\indextraprocessorty
postamble        "\n\n\n\\end{theindextra}\n"

group_skip       "\n\n\n  \\indextraspace"

heading_prefix   "\n\n\n  \\indextragroup{"
heading_suffix   "}\n"
headings_flag    1

item_0           "\n  \\indextraentry{0}{"
item_1           "\n    \\indextraentry{1}{"
item_01           "\n    \\indextraentry{1}{"
item_x1           "}{}\n    \\indextraentry{1}{"
item_2           "\n      \\indextraentry{2}{"
item_12           "\n    \\indextraentry{2}{"
item_x2           "}{}\n    \\indextraentry{2}{"

delim_0          "}{"
delim_1          "}{"
delim_2          "}{"
delim_t          "}"

encap_prefix     "\\"
encap_infix      "{"
encap_suffix     "}"
```

⟨/ist⟩

10.11.2 `.xdy` – `xindy`

⟨∗xdy⟩

```
(markup-index :open  "\begin{theindextra}~n  \indextrastyleversion{0.1}~n  \indextraprocessor
              :close "~n~n\end{theindextra}~n"
              :tree)

(markup-letter-group-list :sep "~n  \indextraspace~n")
(markup-letter-group :open-head "~n  % "
                     :close-head "~n"
                     :group "default")
(markup-letter-group :open-head "~n  \indextragroup{"
                     :close-head "}~n~n")

(markup-indexentry :open "  \indextraentry{0}{"
                   :close "}%~n"
                   :depth 0)
(markup-indexentry :open "}%~n    \indextraentry{1}{"
                   :close ""
                   :depth 1)
(markup-indexentry :open "}%~n      \indextraentry{2}{"
```

```
                    :close ""
                    :depth 2)

(markup-keyword-list :open "" :close "}{" :sep ";")

(markup-locclass-list :open "" :sep ", " :close "")
(markup-locref-list :open ""  :sep ", " :close "")

(markup-range :sep "--")

(markup-crossref-list :class "see"
                      :open "\see{"
                      :sep "}{}, \see{"
                      :close "}{}")
(markup-crossref-list :class "seealso"
                      :open "\seealso{"
                      :sep "}{}, \seealso{"
                      :close "}{}")
```

⟨/xdy⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

K

L

M

N

O

P