The corridx package [*]

Johannes Fink
fink@unileoben.ac.at

May 18, 2005

Abstract

The package introduces a sort entry for chemical names that are preceded by numbers and other prefixes to ensure a correct ordering in the index.

1 Introduction

Chemical Names often have preceding numbers or other prefiexes that do not contribute to the alphabetic sorting of the respective index entry. This drawback can be circumvented by introducing sort keys into the index entry. Such as `\index{dimethylbenzene@1,3-Dimethylbenzene}`. The *corridx* package does this automatically.

Previously a Pascal program entitled *CorrIdx* was used as a preprocessor for the *makeindex* program to insert these key entries into the index entry.

Now a LaTeX macro package is available. This package includes all the necessary modifications directly in the `*.idx` file.

1.1 Clashes

The package `corridx` has a different approach in comparison to the package `index` and will not work with this package and maybe also with other similar packages. A package error is forced when the package `index` is loaded. The package works together with `makeidx`.

2 Usage

2.1 Predefined Index Types

There are always the forms `\ia` and `\noia`, `\ib` and `\noib`, `\ic` and `\noic`, `\ig` and `\noig`. The `\no..` form places an index in the `*.idx` file, whereas the `\i.` form places both text and index.

`\ia` The command `\ia` (index acronym) is used as `\ia{1,2-butanediol}{1,2-BD}` and places in the text "1,2-butanediol (1,2-BD)" and in the index `*.idx`
`\indexentry{acr bd@1,2-BD!\nopagebreak 1,2-Butanediol}{1}`

`\ic` The command `\ic` is used as `\ic{1,2-butanediol}` and places in the text

[*] This document corresponds to corridx v1.0, dated 2005/05/05.

"1,2-butanediol" and in the index *.idx
\indexentry{chem butanediol@1,2-Butanediol}{1}.

\ib The command \ib (**both** acronym and chemical index) is used as \ib{1,2-butanediol}{1,2-BD} and places in the text "1,2-butanediol (1,2-BD)" and in the index *.idx

```
\indexentry{chem butanediol@\begin{crrdxchem}1,2-Butanediol\end{crrdxachem}}{1}
\indexentry{acr bd@1,2-BD\nopagebreak!\begin{crrdxacr}1,2-Butanediol\end{crrdxacr}}{1}
```

\ig \ig (index general) places both in the index and the text, The macro has two arguments, the first being optional. The first optional argument is appended in the index to the second and is suppressed in the text for \ig. \ig[!unsaturated]{polyester} places in the text "polyester" and in the index \indexentry{gen polyester@Polyester!unsaturated}{2}. To use \ig with one argument, \ig{polyester} places places in the text "polyester" and in the index *.idx \indexentry{gen polyester@Polyester}{2}

2.2 Define Yourself an Index Entry Type

The index entry is structured by several variables to have freedom in the output. A typical definition of an index looks like

```
\newcommand{\myindex}[x]{%
\bgroup%
\def\crrdx@key@prefix{keyprefix}%
% here is the key inserted
\def\crrdx@key@ostfix{keypostfix}%
\def\crrdx@sep@@{@separator@}%
\def\crrdx@entry@prefix{entryprefix}%
% here is the entry inserted,
% with the first letter capitalized
\def\crrdx@entry@postfix{entrypostfix}%
\def\crrdx@fmt@page{formatpage}%
\def\crrdx@sfc@prerun{\crrdx@sfc@prerun@default}
\def\crrdx@key@prerun{\crrdx@key@prerun@default}
\crrdx@fmt@index@entry{thekey}{thecapitalizedentry}%
\egroup%
```

In the foregoing definition we give the variables in the order as they are placed by the macro \crrdx@fmt@index@entry. There are more macros to be defined by the user, as are really necessary in most cases, but this kind of structuring may be helpful to achieve the desired result more easily.

Look for the definitions of \noia, \noic, \noig to have a feeling how to use the variables.

2.3 Auxiliary Commands

\crrdxformatpage This macro sets a postfix in the index entry for page formatting. Use the command
\crrdxformatpage{chem}{|textit}

to get in the chemical index the page number in italic. To make the page number for a chemical index in boldface, place the command `\crrdxformatpage{chem}{|textbf}`. This holds for all subsequent entries, if placed in the main text. However, you can redefine the environment `figure`, by including the `\crrdxformatpage` command to get a special page formatting only there.

For example, the definition

```
\renewenvironment{figure}%
{\@float{figure}\crrdxformatpage{chem}{|textit}}%
{\end@float}%
```

causes chemical index entries to be *italic* only in the figure environment, of course, if switched off outside. Do not use `\ic` within the caption, but use `\noic` outside the caption.

Similarly use the commands
`\crrdxformatpage{acr}{}` and
`\crrdxformatpage{gen}{}`, or
`\crrdxformatpage{gen}{|textbf}`.

You can also use your own defined formats.

`crrdxenvchem`
`crrdxenvacr`

We define empty environments that can be later used for special settings in the index. For example, you can define some characters as active there. In my personal copy I use to define this environment `\selectlanguage{chemical}`. The `chemical` package is not available from the LaTeX mirrors. You can download it from `http://www.unileoben.ac.at/~fink/papers/welcome.html` somewhat downwards, in the LaTeXsection.

2.4 Index Sectioning

`\sectioncrrdx`

This macro can be used to place an index heading that is included without numbering into the table of contents. Example:

```
\index{acr␣␣@\sectioncrrdx{Acronyms}\swallow|swallow}%
\index{chem␣␣@\sectioncrrdx{Chemicals}\swallow|swallow}%
\index{gen␣␣@\sectioncrrdx{General}\swallow|swallow}%
```

`\swallow` ensures that no comma and page number will come into the entry. Take care to place the section entry at the top of the other entries, by placing one space more behind `chem␣␣`, etc. than in the other related entries.

3 Implementation

3.1 Main Entries

`\ic`

`\ic{1,2-chemname}` places `1,2-chemname` in the text and places an index entry, formatted as `chem chemname@1,2-chename`.

```
\newcommand{\ic}[1]{\begin{crrdxchem}#1\end{crrdxchem}\noic{#1}}%
```

`\noic`

`\noic{1,2-chemname}` places an index entry, formatted as `chem chemname@1,2-chename`. We encapsule the whole macro by grouping.

```
\newcommand{\noic}[1]{%
\bgroup%
\def\crrdx@key@prefix{chem }%
\def\crrdx@key@ostfix{}%
\def\crrdx@sep@@{@}%
\def\crrdx@entry@prefix{\string\begin{crrdxchem}}%
\def\crrdx@entry@postfix{\string\end{crrdxchem}}%
\def\crrdx@fmt@page{\crrdx@fmtpage@chem}%
\def\crrdx@sfc@prerun{\crrdx@sfc@prerun@default}%
\def\crrdx@key@prerun{\crrdx@key@prerun@default}%
\crrdx@fmt@index@entry{#1}{#1}%
\egroup%
}%
```

`\ia` This is the acronym entry. It places the acronym in the text and in the index

```
\newcommand{\ia}[2]{#1 \mbox{(#2)}\noia{#1}{#2}}%
```

`\noia` This places the acronym in the index.

```
%
\newcommand{\noia}[2]{%
\bgroup%
\def\crrdx@key@prefix{acr }%
\def\crrdx@key@ostfix{}%
\def\crrdx@sep@@{@}%
\def\crrdx@entry@prefix{#2\string\nopagebreak!\string\begin{crrdxacr}}%
\def\crrdx@entry@postfix{\string\end{crrdxacr}}%
\def\crrdx@fmt@page{\crrdx@fmtpage@acr}%
\def\crrdx@sfc@prerun{\crrdx@sfc@prerun@default}%
\def\crrdx@key@prerun{\crrdx@key@prerun@default}%
\crrdx@fmt@index@entry{#2}{#1}%
\egroup%
}%
```

`\ib \noib` `\ib` and `\noib` are macros to place both a chemical index and an acronym. `\ib` places both the index and the text, whereas `\noib` places only both index entries. The macros have two arguments, e.g., `\ib{1,2-chemname}{CN}`.

```
\newcommand{\noib}[2]{\noic{#1}\noia{#1}{#2}}%
\newcommand{\ib}[2]{\ic{#1}\noia{#1}{#2}}%
```

`\ig` and `\noig` are macros to place a general index.

`\ig`

```
\newcommand{\ig}[2][]{%
#2%
\noig[#1]{#2}%
}%
```

`\noig` This macro formats a general index entry.

```
\newcommand{\noig}[2][]{%
\bgroup%
\def\crrdx@key@prefix{gen }%
\def\crrdx@key@ostfix{}%
\def\crrdx@sep@@{@}%
\def\crrdx@entry@prefix{}%
\def\crrdx@entry@postfix{#1}%
\def\crrdx@fmt@page{\crrdx@fmtpage@gen}%
\def\crrdx@sfc@prerun{\crrdx@sfc@prerun@default}%
\def\crrdx@key@prerun{\crrdx@key@prerun@default}%
\crrdx@fmt@index@entry{#2}{#2}%
\egroup%
}%
```

3.2 Auxiliary Commands

\crrdxformatpage

This macro set a postfix for page formatting.

```
\newcommand{\crrdxformatpage}[2]{\@namedef{crrdx@fmtpage@#1}{#2}}%
\crrdxformatpage{chem}{\@empty}%
\crrdxformatpage{acr}{\@empty}%
\crrdxformatpage{gen}{\@empty}%
```

The `makeindex` program describes the following commands for page formatting:

```
\providecommand{\ii}[1]{{\it#1}}%
\providecommand{\bb}[1]{{\bf#1}}%
```

e.g., for index boldface pagenumber as last entry in the `\index` command `|bb`. However we not use this type, we use rather `|textit}` or `|textbf`.

crrdxchem

We define an empty environment that can be later used for special chemical language settings.

```
\newenvironment{crrdxchem}{}{}%
```

crrdxacr

We define an empty environment that can be later used for special settings for acronyms.

```
\newenvironment{crrdxacr}{}{}%
```

3.3 Formatting the Key

\crrdx@key

This macro scans the entry and removes every non-letter character. The key is placed in the index before the `@` as sorting aid.

```
\newcommand{\crrdx@key}[1]{%
\crrdx@key@prerun%
\edef\crrdx@key@temp{#1}%
\edef\crrdx@key@var{\@empty}%
\let\end\@empty%
\expandafter\crrdx@key@loop\crrdx@key@temp\end%
}%
```

```
\newcommand{\crrdx@key@loop}[1]{%
\let\next=\crrdx@key@loop%
\ifx#1\end\let\next=\relax\fi%
\ifcat\noexpand#1a\lowercase{\edef\crrdx@key@var{\crrdx@key@var#1}}\fi%
\next}%
```

The following text formats are skipped in the key:

```
\newcommand{\crrdx@key@prerun@default}%
{%
\renewcommand{\textrm}[1]{{\noexpand\textrm{##1}}}%
\renewcommand{\textbf}[1]{{\noexpand\textbf{##1}}}%
\renewcommand{\textsf}[1]{{\noexpand\textsf{##1}}}%
\renewcommand{\texttt}[1]{{\noexpand\texttt{##1}}}%
\renewcommand{\textmd}[1]{{\noexpand\textmd{##1}}}%
\renewcommand{\textit}[1]{{\noexpand\textit{##1}}}%
\renewcommand{\textsc}[1]{{\noexpand\textsc{##1}}}%
\renewcommand{\textsl}[1]{{\noexpand\textsl{##1}}}%
\renewcommand{\textup}[1]{{\noexpand\textup{##1}}}%
\def~{\string~}%
}%
```

If you want to skip a normal text in the key, then format it as `\textup`.

3.4 Formatting the Entry

`\crrdx@sfc@prerun` After scanning the key, we want to use the formats again with a different expansion than when doing the key.

```
\newcommand{\crrdx@sfc@prerun@default}%
{%
\renewcommand{\textrm}{\string\textrm}%
\renewcommand{\textbf}{\string\textbf}%
\renewcommand{\textsf}{\string\textsf}%
\renewcommand{\texttt}{\string\texttt}%
\renewcommand{\textmd}{\string\textmd}%
\renewcommand{\textit}{\string\textit}%
\renewcommand{\textsc}{\string\textsc}%
\renewcommand{\textsl}{\string\textsl}%
\renewcommand{\textup}{\string\textup}%
\def~{\string~}%
}%
```

`\crrdx@fmt@index@entry` This macro formats the input into the index.

```
\newcommand{\crrdx@fmt@index@entry}[2]{%
\crrdx@key{#1}%
\crrdx@sfc{#2}%
\edef\crrdx@fmt@index@entry@var{%
\crrdx@key@prefix%
\crrdx@key@var%
\crrdx@key@ostfix%
\crrdx@sep@@%
\crrdx@entry@prefix%
\crrdx@sfc@var%
\crrdx@entry@postfix%
\crrdx@fmt@page}%
\index{\crrdx@fmt@index@entry@var}%
}%
```

`\crrdx@sfc@loop` We want to capitalize the first letter automatically. We start to declare the loop used in `\crrdx@sfc`. We scan the entry to find the first letter. This we are capitalizing and then we switch to scan what is left over from `#1`.

```
\newcommand{\crrdx@sfc@loop}[1]{%
\crrdx@sfc@prerun%
\let\next=\crrdx@sfc@loop%
\ifx#1\end\let\next=\relax\fi%
\ifcat\noexpand#1a%
```

A letter is found

```
\uppercase{\edef\crrdx@sfc@var{\crrdx@sfc@var#1}}%
```

make it uppercase. We scan now the rest what is left over in `#1` and add it to the `\crrdx@sfc@var`.

```
\let\next=\crrdx@sfc@loop@leftover%
\else%
\edef\crrdx@sfc@var{\crrdx@sfc@var#1}%
\fi%
\next}%
```

`\crrdx@sfc@loop@leftover` Collects the stack `#1` after finding a letter.

```
\newcommand{\crrdx@sfc@loop@leftover}[1]{%
\let\next=\crrdx@sfc@loop@leftover%
\ifx#1\end\let\next=\relax\fi%
\edef\crrdx@sfc@var{\crrdx@sfc@var#1}%
\next%
}%
```

`\crrdx@sfc` This is the main macro to **S**et the **F**irst letter as **C**apital, i.e., what is behind the `@` in the `\index` command. We presume that before a possible space in the entry there is a letter that can be capitalized.

```
\newcommand{\crrdx@sfc}[1]{%
\edef\crrdx@sfc@var{\@empty}%
\let\end\@empty%
\crrdx@split{#1}%
\expandafter\crrdx@sfc@loop\crrdx@beforespace\end%
\ifx\crrdx@afterspace\@empty\else%
\edef\crrdx@sfc@var{\crrdx@sfc@var\space\crrdx@afterspace}%
\fi%
}%
```

Scanning the loop, normally spaces are lost. Therefore we split the entry when a space is there at the first space and glue it together in the final part of `\crrdx@sfc`.

```
\def\crrdx@ss#1 #2\stop{\edef\crrdx@beforespace{#1}\edef\crrdx@afterspace{#2}}%
```

```
\newcommand{\crrdx@split}[1]{%
\edef\test{\@empty}%
```

We test for no space there with a space and `\test` as last entry.

```
\crrdx@ss#1 \test\stop%
\ifx\crrdx@afterspace\test%
```

there is no space

```
\edef\crrdx@beforespace{#1}\edef\crrdx@afterspace{\@empty}%
\else%
```

there is a space

```
\crrdx@ss#1\stop%
\fi%
}%
```

3.5 Index Sectioning

We use here `providecommand` because it may be defined in some classes too (at least in classes that I developed for personal use).

`\swallow` Swallows the next item (no operation)

```
143 \providecommand\swallow[1]{\relax}%
```

`\sectioncrrdx` Makes an index, added to table of contents at level 1 (section) without number.

```
144 \providecommand{\sectioncrrdx}[1]%
145 {\section*{#1}\addcontentsline{toc}{section}{#1}}%
```

3.6 Clashes Check

```
146 \AtBeginDocument{ \@ifpackageloaded{index}%
147 {\PackageError{corridx}%
148 {This package is not compatible with package index}{}}}%
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.