XeIndex

v.0.3 – 2014/12/05

(I simply added a missing comment sign after 4 years of inactivity.
The problem was spotted by Tomas Jonsson.)

Paul Isambert – zappathustra@free.fr

Abstract

XeIndex is a package based on XeSearch that automatically indexes words in a XeLaTeX document. Words or phrases (possibly underspecified) are declared in a list and each of their occurrences then creates an index entry, whose content might be freely specified beforehand.

Automatic indexes are bad. You know that. Hence the severe look of this documentation. So: don't use XeIndex. Or: use it as a tool to generate an index whose relevance you then check. Or: use it for entries that are generally indexed on every occurrence, like proper names. Or: do bad indexes.

You load XeIndex in the usual way:

```
\usepackage{xeindex}
```

There's only one package option, namely `mark`, which prefixes all index entries generated by XeIndex with `***[`⟨*word*⟩`:`⟨*line number*⟩`]`, where ⟨*word*⟩ is the word that generated the entry and ⟨*line number*⟩ the line where you can find it in the `.tex` file. Moreover, if a word appears more than once on a page, and thus generates several index entries although they'll be merged in the typeset index, with this option each entry is listed along with each occurrence of the word. So these entries end up at the beginning of the index, sorted with symbols, apart from the entries generated by the usual `\index` command. Thus they can be easily checked and prevented if irrelevant.

`\makeindex`
`\printindex`
`\index{`⟨entry⟩`}`

XeIndex loads the `makeidx` package, so `\makeindex` and `\printindex`, which should be executed as usual, as well as the usual `\index` command, are available. (If you don't know what I'm talking about, then you don't know how to produce an index. You should read the `makeidx` documentation at least, or the rest of this document might seem cryptic.)

`\IndexList`⟨`*`⟩`{`⟨`name`⟩`}{`⟨`list of entries`⟩`}`

This is XeIndex's main command. The star is optional and ⟨*list of entries*⟩ is made of ⟨*word*⟩`=`⟨*entry*⟩ pairs separated by commas, with the part in red optional too. Let's see the simplest possibility first:

```
\IndexList{mylist}{alley cat,dog,gnu}
```

This will index *alley cat* on every occurrence of *alley cat* in your document, *dog* on every occurrence of *dog*, and *gnu* on every occurrence of *gnu*. But this will also index *alley cat* when seeing *Alley cat*, *dog*

when seeing *dOg*, and *gnu* when seeing *GNU*. In other words, XɜIndex is case-insensitive by default, and the index entries are put in lower case. Most of the time, that's useful. But a gnu is not GNU, so sometimes you want a different behavior. In this case, put a * before the word, e.g.:

```
\IndexList{mylist}{alley cat,DOG,gnu,*GNU}
```

Now XɜIndex will index *gnu* when seeing *gnu* or *Gnu*, but it will index *GNU* when seeing *GNU*. Note that `DOG` without a star will index *dog* in any case display, and the index entry will be *dog*, not *DOG*. You can also make a whole list case-sensitive, by adding the star just after `\IndexList`, as in:

```
\IndexList{mylist}{alley cat,dog,gnu}
\IndexList*{mylist}{GNU}
```

All words in a *-marked list are case-sensitive, and you should not add another star before them. Note that you can use the same list with different case-sensitivies each time, as in the above example.

Case-sensitity is useful mostly to index proper names.

The words in a list need not be fully specified. You can index all words beginning or ending with some letters. To do so, use ? for the unspecified part. For instance,

```
\IndexList{mylist}{cat?,?mals,*?NU}
```

will find and index *cat*, *cats*, *catheter*, *animals*, *mammals*, *GNU*, *gNU*, but not *gnu*, because `*?NU` means `a word that ends with ``NU'' in upper case.' Each of these words will be indexed in its own entry, so there'll be a *cat* and a *cats* entry, but we'll learn how to have them grouped presently. If a word matches several underspecified forms, the more specific wins, e.g. *mammals* matches `mam?` and not `ma?`, and `starting-with' forms always win against `ending-with' ones, no matter the degree of specificity, e.g. *cat* matches `c?` and not `?at`. Phrases, i.e. words separated by blanks (like *alley cat* above) can be underspecified only at the end, i.e. you can say `alley ca?` but not `?ley cat` (actually you can try but it won't work). Finally, underspecified parts are only the beginning or the end of a word, i.e. you can't say `c?t`, and there should be only one underspecification, i.e. `?a?` is forbidden.

Now we come to the interesting part in ⟨*word*⟩=⟨*entry*⟩, namely the red part: ⟨*entry*⟩ should be a pseudo-index entry, as it were, i.e. it can be a normal index entry, as in

```
\IndexList{another list}{%
  dog=mammal|textit,
  snail=Obviously Not Mammal,
  cat=mammal!pet,
```

```
  horse?=horse@{\bfseries horse}}
```

Here, every occurrence of *dog* will produce an index entry at *mammal* with the page number in italic, *snail* will index *Obviously Not Mammal*, *cat* will create a sub-entry *pet* in the *mammal* entry and *horse*, *horses*, *horseshoe*, etc., will create a bold entry at *horse*. So, as you can see, the left part of the expression specifies which words should generate an index entry, and the right part is the index entry that should be generated. The latter is not case-insensitive anymore: although `snail` is case-insensitive and will fire on *snail*, *Snail* or *SNaIl*, the index entry itself will be as specified, i.e. *Obviously Not Mammal* and not *obviously not mammal*.

Now you can index variations of a word under the same entry. As I've just said, *horse* and *horses* will both be indexed under *horse*, as well as all their case-variants, as we already know. (*Horseshoe* will go in that entry too, so beware).

The right part of those pairs is a `pseudo-index entry,' and not a proper index entry, because the word to index can be omitted, in which case the word that fires the index entry will be used instead. More precisely, whenever XǝIndex encounters one of the usual MakeIndex operators, namely `!` (for a sub-entry), `|` (for a control sequence) and `@` (to indicate the form of the entry irrespective of alphabetical order), at the beginning of an index entry, it reinterprets this entry with the the firing word or phrase on its left. I.e.

```
\IndexList{yet another list}{%
  dog=@dog (canis lupus familiaris),
  cat=!basic definition,
  horse?=|textbf}
```

indexes every occurence of *dog* as *dog (canis lupus familiaris)*, but with the alphabetical order of *dog*, every occurrence of *cat* as a *basic definition* subentry in the *cat* entry, and every occurrence of a word beginning with *horse* as an entry for this word with the page number in boldface. Remember that the `\index` command is still available, so you can still index *cat* as *cat* in your document.

It's obviously a very bad idea to say something like `\IndexList{mylist}{cat=|(}`, since this will fire `\index{cat|(}` on every occurrence of *cat*. So page ranges can't be declared by XǝIndex (it would be a very bad idea anyway). On the other hand, MakeIndex automatically creates page ranges as soon as an entry is found on at least three successive pages, unless you run it with the `-r` option.

If you want a comma in the right part of the entry, enclose the entire entry, minus the MakeIndex operator if any, between braces, e.g.:

```
\IndexList{writers}{Kafka={Kafka, Franz}}
```

(This did not work in version 0.1, and now it's corrected thanks to Simon Spiegel who indicated it to me.)

\StopIndexList{⟨lists⟩} Here are some additional macros to let you regulate the flow of the indexing frenzy. \StopIndexList takes a comma-separated list of lists and turn them off. \StopIndex turns off all lists. \NoIndex simply prevents ⟨*text*⟩ from being indexed. It's very important, because it lets you prevent irrelevant indexation in the body of your document.

\StopIndex

\NoIndex{⟨text⟩}

That's all you need to know to use XǝIndex. The next paragraph describes how XǝIndex sets the parameters of XǝSearch; so if you don't know XǝSearch and don't intend to use it, there's no need to read what follows.

XǝIndex keeps XǝSearch's default search order, namely full words before prefixes before suffixes, with case-sensitive tests first each time. Affixes are modified, however: they're sorted by length (longer ones first) and not kept in the order they were declared, and only one affix fires in case of a successful test, instead of all the affixes of a given test. You can modify these specifications since XǝIndex uses `!`-marked replacement texts, so they won't embed each other, but then you might end up with multiple entries and a lack of consistency.

You can use \StartSearching and \StopSearching instead of \StopIndex, which for the moment renders all lists unavailable. The former two commands, however, will stop all lists defined by XǝSearch.

The default set of boundaries is left untouched, i.e. its members are: `.,;:-`'()[]{}`

Implementation

Basic declarations and definitions.

```
\ProvidesPackage{!FileName}[!FileDate !FileVersion Automatic index for XeLaTeX.]
\RequirePackage{makeidx,XǝSearch}
\makeatletter
\newif\ifxi@mark
\DeclareOption{mark}{\xi@marktrue}
\ProcessOptions
```

\xi@Mark \xi@empty \xi@end \xi@Lists

\xi@Mark either shows the word and the corresponding line, or it gobbles it. It is placed at the beginning of the \index command, whose expansion is delayed accordingly.

```
\ifxi@mark
  \def\xi@Mark#1{***[#1:\the\inputlineno] }
```

```
\else
  \def\xi@Mark#1{}
\fi
\def\xi@empty{}
\def\xi@end{\xi@end}
\def\xi@Lists{}
```

\IndexList
\xi@IndexList

Most of the job is done by XɘSearch. What we need to do is properly analyze the entry to launch an adequate search.

```
\def\IndexList{%
  \@ifstar{\def\xi@cs{*}\xi@IndexList}{\def\xi@cs{}\xi@IndexList}%
  }
\def\xi@IndexList#1#2{%
  \def\xi@ListName{#1}%
  \edef\xi@Lists{\xi@Lists#1,}%
  \unless\ifcsname#1@xeindex\endcsname
    \csname#1@xeindex\endcsname
```

When a index list is created, we associate five XɘSearch lists with it: one is for words and affixes that should index themselves in lower case.

```
    \SearchList{#1@xeindex@ncs@normal@list}{%
      \def\xi@Word{##1}%
      \lowercase{\expandafter\index\expandafter{\xi@Mark\xi@Word##1}}}{}%
```

Another one is for case-sensitive words and affixes.

```
    \SearchList{#1@xeindex@cs@normal@list}{%
      \expandafter\index\expandafter{\xi@Mark{##1}##1}}{}%
```

And the last three are for words and affixes that launch a special entry, which is stored in an associated command.

```
    \SearchList{#1@xeindex@ncs@special@list}{%
      \lowercase{\csname##1@#1@xeindex@entry\endcsname}{##1}}{}%
    \SearchList{#1@xeindex@cs@special@list}{%
      \csname##1@#1@xeindex@entry\endcsname{##1}}{}%
    \SearchList{#1@xeindex@affix@special@list}{%
      \csname\AffixFound @#1@xeindex@entry\endcsname{##1}}{}%
  \fi
  \xi@ParseList#2,\xi@end,%
```

```
}
```

`\xi@ParseList` This macro recursively tests each entry in `\SearchList` and feed it to `\xi@ParseEntry` with an additional `=` to check for the right part. It also adds `\xi@cs`, which was defined do `*` in case `\SearchList` was starred.

```
\def\xi@ParseList#1,{%
  \def\xi@temp{#1}%
  \ifx\xi@temp\xi@end
    \let\xi@next\relax
  \else
    \expandafter\xi@ParseEntry\xi@cs#1=\xi@end
    \let\xi@next\xi@ParseList
  \fi\xi@next
  }
```

`\xi@ParseEntry` This one analyses the entry. If the third argument is empty, then there is no $=\langle entry \rangle$ part in the entry. In this case we add the word or affix to one of the two simple lists, depending on its case-sensitivity.

```
\def\xi@ParseEntry#1#2=#3\xi@end{%
  \def\xi@temp{#3}%
  \ifx\xi@temp\xi@empty
    \expandafter\if\noexpand#1*%
      \AddToList!{\xi@ListName @xeindex@cs@normal@list}{#1#2}%
    \else
      \AddToList!{\xi@ListName @xeindex@ncs@normal@list}{#1#2}%
    \fi
```

Otherwise we feed the right part to `\xi@MakeEntry` which sets the `\ifxi@NoWord` switch. We also check whether the word is an affix or not, and whether it is case-sensitive. The word is then associated to the right list and an associated macro is created.

```
  \else
    \xi@MakeEntry#3%
    \expandafter\if\noexpand#1*%
      \xi@CheckAffix#2?\xi@end
      \ifxi@Affix
          \AddToList!{\xi@ListName @xeindex@affix@special@list}{#1#2}%
          \expandafter\edef\csname\xi@Affix @\xi@ListName @xeindex@entry\endcsname##1{%
```

```
          \unexpanded{\expandafter\index\expandafter}{%
            \noexpand\xi@Mark{##1}\ifxi@NoWord##1\fi\unexpanded\expandafter{\xi@temp}}%
          }%
    \else
      \AddToList!{\xi@ListName @xeindex@cs@special@list}{#1#2}%
        \expandafter\edef\csname#2@\xi@ListName @xeindex@entry\endcsname##1{%
          \unexpanded{\expandafter\index\expandafter}{%
            \noexpand\xi@Mark{##1}\ifxi@NoWord#2\fi\unexpanded\expandafter{\xi@temp}}%
          }%
    \fi
  \else
    \xi@CheckAffix#1#2?\xi@end
    \ifxi@Affix
        \AddToList!{\xi@ListName @xeindex@affix@special@list}{#1#2}%
        \expandafter\edef\csname\xi@lcAffix @\xi@ListName @xeindex@entry\endcsname##1{%
          \unexpanded{\def\xi@Word}{##1}%
          \noexpand\lowercase{%
            \unexpanded{\expandafter\index\expandafter}{%
              \unexpanded{\xi@Mark{\xi@Word}}%
              \ifxi@NoWord##1\fi\unexpanded\expandafter{\xi@temp}}%
            }%
          }%
    \else
      \AddToList!{\xi@ListName @xeindex@ncs@special@list}{#1#2}%
      \lowercase{%
        \expandafter\edef\csname#1#2@\xi@ListName @xeindex@entry\endcsname##1{%
          \unexpanded{\def\xi@Word}{##1}%
          \unexpanded{\expandafter\index\expandafter}{%
            \unexpanded{\xi@Mark{\xi@Word}}%
            \ifxi@NoWord#1#2\fi\unexpanded\expandafter{\xi@temp}}%
          }%
        }%
    \fi
  \fi
```

```
  \fi
  }
```

`\xi@MakeEntry` This determines whether the entry starts with one of the MakeIndex operators.

```
\newif\ifxi@NoWord
\def\xi@exclam{!} \def\xi@at{@} \def\xi@bar{|}
\def\xi@MakeEntry#1#2={%
  \def\xi@temp{#1#2}%
  \xi@NoWordtrue
  \unless\ifx\xi@temp\xi@exclam
    \unless\ifx\xi@temp\xi@at
      \unless\ifx\xi@temp\xi@bar
        \xi@NoWordfalse
      \fi
    \fi
  \fi
  }
```

`\xi@CheckAffix` If the first argument is `?`, then the word is unspecified at the beginning. Otherwise, if the third argument is not empty, then it is unspecified at the end (because we added a `?` when giving the word to this macro). In case the `?` is misplaced, XɆSearch will detect it later.

```
\newif\ifxi@Affix
\def\xi@CheckAffix#1#2?#3\xi@end{%
  \xi@Affixfalse
  \expandafter\if\noexpand#1?%
    \xi@Affixtrue
    \def\xi@Affix{#2}%
    \lowercase{\def\xi@lcAffix{#2}}%
  \else
    \def\xi@@temp{#3}%
    \unless\ifx\xi@@temp\xi@empty
      \xi@Affixtrue
      \def\xi@Affix{#1#2}%
      \lowercase{\def\xi@lcAffix{#1#2}}%
    \fi
  \fi
```

```
  }
```

\StopIndexList These are straightforward.

\xi@StopIndexList
\StopIndex
\NoIndex

```
\def\StopIndexList#1{%
  \xi@StopIndexList#1,\xi@end,%
  }%
\def\xi@StopIndexList#1,{%
  \def\xi@temp{#1}%
  \ifx\xi@temp\xi@end
    \let\xi@next\relax
  \else
    \StopList{%
      #1@xeindex@ncs@normal@list,%
      #1@xeindex@cs@normal@list,%
      #1@xeindex@cs@normal@list,%
      #1@xeindex@ncs@special@list,%
      #1@xeindex@cs@special@list,%
      #1@xeindex@affix@special@list%
      }%
    \let\xi@next\xi@StopIndexList
  \fi\xi@next
  }
\def\StopIndex{%
  \expandafter\xi@StopIndexList\xi@Lists\xi@end,%
  }
\def\NoIndex#1{%
  \bgroup
  \StopIndex
  #1%
  \egroup
  }
```

\xi@PrintIndex Finally, we patch \printindex so it won't be searched, and sets XƎSearch's parameters.

```
\let\xi@PrintIndex\printindex \def\printindex{\StopIndex\xi@PrintIndex}
\SortByLength{pPsS} \SearchOnlyOne{pPsS}
\makeatother
```