esindex

Generating automatically sort keys for *MakeIndex* with LaTeX

Version 1.8 (2024-03-27)

`https://github.com/jbezos/esindex`

Javier Bezos (`http://www.texnia.com`)

Please, report any issues you find in `https://github.com/jbezos/esindex/issues`.

This package defines the command `\esindex`, which adds sort keys to the index entries for you. Although originally a specifically Spanish tool (so part of the documentation is in Spanish), the idea behind this package may be applied to other languages, and tools to adapt it are provided. They are explained below, section 2.

So,

```
\esindex{cañón}
```

is equivalent to

```
\index{can^^ffon@cañón}
```

`\index` is not touched at all (except if requested with the package option `replaceindex`).

As you can see, the package generates the sort key within TeX itself, which has a number of advantages (and some disadvantages, of course).

Version 1.5 provides tools for other languages and adds support for luatex and xetex.

Version 1.6 adds a package option – with `babel` the 'actual' value (ie, the text to be printed in the index) is wrapped with `\foreignlanguage` if the current language is not the main one (this replacement comes after `\esindexactual`). Actually, the macro is `\esindexlanguage`, which by default is 'let' to the babel macro.

Version 1.7 (besides fixing a bug with xe/lua) adds multilevel comparisons (up to three levels, ie, primary, secondary and tertiary). For example:

```
\esindexreplacesub{å}{a}{2}%
\esindexreplacesub{à}{a}{1}%
```

creates a two-level key sorting à before å. For a three-level key use `\esindexreplacesubsub` (with 4 arguments). With `\esindexexpandkeys` the three levels are expanded. (Internally, with the definitions above the key for `ràpid` is `rapid^^Ar1pid`.)

(Version 1.8 is just a bug fix.)

1. Spanish

Este paquete ha sido diseñado para facilitar la escritura de índices correctamente alfabetizados en castellano. Su principal orden es `\esindex`, que convierte a una forma adecuada su argumento. Así por ejemplo,

```
\esindex{cañón}
```

equivale a

```
\index{can^^ffon@cañón}
```

No es necesario usar babel salvo, lógicamente, si los acentos están escritos en forma de abreviaciones (`'a`, `'e`, etc.) en lugar de con los caracteres reales. En este último caso, el paquete utiliza ciertas órdenes internas de babel, por lo que no puedo garantizar su funcionamiento correcto con versiones distintas a las 3.6 a 3.27. En caso de que esindex sea incompatible con futuras versiones de babel (lo cual no es realmente probable) intentaré adaptarlo en el menor tiempo posible.

Salvo el carácter `actual` (normalmente `@`) se pueden usar todos los caracteres especiales de *MakeIndex*. Se pueden aplicar convenciones diferentes a las normales, pero en este caso hacen falta ajustes adicionales en caso de que los modificados sean `actual`, `encap`, `level` o `quote`. En ese caso basta con indicar los caracteres que hay que usar como opciones de paquete. Por ejemplo, si para `quote` decidimos usar `$` en nuestro archivo `.ist` particular, tendríamos que llamar al paquete del siguiente modo:

```
\usepackage[quote=$]{esindex}
```

Es importante observar que, a diferencia de la opción para alemán de *MakeIndex*, el uso de `"` en abreviaciones como `"u` es completamente legítimo, ya que el paquete reconoce tal combinación y la trata aparte. Lo mismo vale para `'` o `~` en caso de que se usaran como carácter especial. Es decir

```
\esindex{{"!'}Cig"ue'nas{"!}|textbf}
```

equivale a

```
\index{{"!'}Ciguen^^ffas{"!}@{"!'}Cig\"ue\~nas{"!}|textbf}
```

Sin embargo, el uso del carácter `quote` ante `encap` o `level` no se detecta a menos que el grupo esté encerrado entre llaves. Por ejemplo, en lugar de `\esindex{Pleca: "|}` debe escribirse `\esindex{Pleca: {"|}}`. (En realidad en este caso podría haberse usado `\index`. Es tan sólo un ejemplo.)

Aunque el hecho de que `@` no se pueda usar en `\esindex` hace que todavía algunas entradas se tengan que hacer a mano, la mayor parte del trabajo se ve considerablemente simplificado.

Finalmente, hay que señalar que con este paquete no se crea en el índice una entrada propia para la palabras que empiezan por eñe, sino que tan sólo se añaden al final de la ene. En el rarísimo caso de que hubiera palabras que empiezan por eñe habría que modificar el archivo `.ind` a mano o bien redefinir de algún modo las entradas generadas.

La versión 1.3 elimina una incompatibilidad con recientes versiones de LaTeX y añade nuevas funciones:

- Opción de paquete `ignorespaces`: al formar la clave de ordenación se suprimen los espacios, de forma que:

$$\text{adentro} < \text{a donde} = \text{adonde}.$$

- Opción de paquete `replaceindex`: el comportamiento de `\index` se reemplaza por el de `\esindex`, aunque en este caso no es posible introducir entradas que no se adapten a lo requerido por `\esindex`.
- La orden `\ignorewords` da una lista de palabras separadas por comas que no se cuentan en la ordenación. Por ejemplo, con `\ignorewords{de}` tendríamos:

  $$\text{pino albar} < \text{pino laricio} < \text{pino de montaña}.$$

  Distingue la caja, por lo que las formas con mayúsculas hay que darlas explícitamente, si hicieran falta.

Algunas funciones adicionales son:

- La lista de tókenes `\everyesindex` permite dar definiciones locales para establecer el comportamiento de otras órdenes. Por ejemplo:

  ```
  \everyesindex{\renewcommand\emph[1]{#1}}
  ```

  elimina de la clave esa orden. Con:

  ```
  \everyesindex{\renewcommand\emph[1]{#1'}}
  ```

  la entrada en cursiva iría detrás de la redonda, si la hubiera. Es una técnica que se puede emplear en otros casos para reajustar el orden de entradas idénticas.
- La orden `\esindexsort` permite predefinir claves asociadas a entradas concretas, para ajustar su ordenación (lo que normalmente se consigue añadiendo texto adicional para que makeindex lo tenga en cuenta). Estas correspondencias deben darse antes de la aparición del primer `\esindex` con ese término, y las claves se procesan posteriormente con `ignorespaces`, `\ignorewords` y `\everyesindex`, si están activadas. Por ejemplo:

  ```
  \esindexsort{adonde}{adonde'1}
  \esindexsort{adónde}{adonde'2}
  \esindexsort{a donde}{a donde'7}
  \esindexsort{a dónde}{a donde'8}
  ```

  daría el orden *adonde, adónde, a donde, a dónde*, con `ignorespaces`, o bien *a donde, a dónde,* [probablemente otros términos], *adonde, adónde*, sin `ignorespaces`.

2. Other languages

First, you very likely want to ignore de Spanish specific settings with the package option `nospanish`. What `\esindex` does is the following:

1. Replacements set by `\esindexsort`. See an example above.
2. `\esindexreplace` in `\everyesindex`. See an example below.
3. Replacements by LaTeX (protected) expansion, including redefinitions in `\everyesindex` and, in Spanish, `\'`, `\"` and `\~`. You may force protected expansion at any point inside `\everyesindex` with `\esindexexpandkey`. In addition, the sort key is stored in `\esindexkey`, which can be manipulated directly.

4. Removal of words listed in `\ignoredwords` (maybe not the logical place, but real life is not always logical). For example, with `\ignorewords{de,la}` the words "de" and "la" (preceded and followed by spaces) are removed from the key.
5. Removal of spaces if `ignorespaces`. Very often this is similar to the `-l` option of *MakeIndex*.
6. Replacement of 'actual' as set by `\esindexactual`, but still based on the original `\esindex` argument, without changes. For example, with:

   ```
   \esindexactual{Felipe II}{Felipe II, \textit{rey de España}}
   ```

   just write `\esindex{Felipe II}` (as many times as you want), and the entry will show "Felipe II, *rey de España*" (the sort key is still based on "Felipe II", of course).

Do you find it chaotic? Well, you are right. After all this package was for Spanish indexes with some readjustments for it to be adapted to other languages. A general solution deserves another package. Feel free to create one based on this package (MIT license), if you like.

Here is an example of an `\everyesindex`. Like other `\every...`'s, it is a token register:

```
\everyesindex{%
  \renewcommand\"[1]{#1e}%
  \renewcommand\textit[1]{#1}%
  \esindexexpandkey
  \esindexreplace{ä}{ae}%
  \esindexreplace{ü}{ue}%
  \esindexreplace{ö}{oe}}
```

What it does is:

1. Redefines `\textit` so that it is removed to build the key. With `\everyesindex{\renewcommand\textit[1]{#1'}}` italics would be sorted after upright.
2. Redefines `\"`. Of course, this only works correctly if used for this precise expansion.
3. Applies the previous changes with a protected expansion.
4. Make a direct replacement of some characters.

Note there is a pattern: first flatten the key by removing macros, and then the resulting unmarked text is processed to replace some chars.

Remember these changes are not shown – they are used by *MakeIndex* to sort the entries. Note also replacements can be made language dependent with the appropriate test (eg, with iflang).

As a convenience tool, `\esindexlastchar` is `^^ff` or `^^^^ffff` (`^^^^^^10ffff` in luatex), depending on the engine.

You can play with lua inside `\everyesindex`, too. For example, to convert at once things like *2,3-dimetilfenol* to *dimetilfenol*:

```
\everyesindex{
  \directlua{
    token.set_macro('esindexkey',
                    string.gsub([[\esindexkey]], '^[-0-9,]+', ''))
  }}
```

It is just the concept (for example, it fails if the key contains `]]`).

3. Final remarks

Indexing has many subtleties. Recommended readings are:

- *Guidelines for Alphabetical Arrangement of Letters and Sorting of Numerals and Other Symbols*,
  `https://www.niso.org/sites/default/files/2017-08/tr03.pdf`
- *Guidelines for Indexes and Related Information Retrieval Devices*,
  `https://www.niso.org/sites/default/files/2017-08/tr02.pdf`
- *Unicode Collation Algorithm*, `https://unicode.org/reports/tr10/`,
- *Unicode Locale Data Markup Language (LDML)*, part 5, *Collation*,
  `http://www.unicode.org/reports/tr35/tr35-collation.html`.

There are at least 3 tasks, from LaTeX to a suitable key (eg, `$\delta$` to `delta` or whatever you want), collating these keys, and generate the formated index in a form LaTeX understands.[1] With makeindex these 3 steps are intermingled, and so does esindex. This is another reason why a general solution deserves a completely new package. However, with some patience you can do many useful things with esindex, even sorting words in non-Latin scripts (at least, with the most basic rules).

[1] There can be also sub-tasks. For example, the last task is best split in two sub-tasks, namely, first generate a sorted list of terms, and then format them, but unfortunately, most `ist` styles output a formatted list. We also need to format and collapse locators and so on.