Forindex: computer-aided indexing

Guido Milanese

Abstract

A good index is a very important tool in writing: it improves the way readers approach books, manuals, and proceedings. Although a certain amount of data must be entered manually, some rather trivial tasks can be performed automatically, e.g. an index of geographical names. For such a purpose, `Forindex` is a standalone utility offering two basic functions: `doindex` prepares a file to be processed by `makeindex`, and `cleanindex` removes all the `\index{...}` entries in a LaTeX file, obtaining a clean file with no index tags: this can be useful if the file is to be indexed from scratch, maybe with different criteria. The program is written in Snobol4, using Zenity as GUI, which makes the code almost 100% portable. For Windows a standalone executable file is also provided.

Sommario

Un indice ben fatto costituisce uno strumento importante nel processo di scrittura: migliora la fruibilità di libri, manuali e atti di convegni. Anche se alcuni dati debbono necessariamente essere introdotti manualmente, compiti piuttosto banali possono compiersi automaticamente, per esempio un indice di nomi geografici. A questo scopo, `Forindex` è un programma standalone che offre due funzioni fondamentali: `doindex` prepara un file per essere processato da `makeindex`, mentre `cleanindex` rimuove tutte le marcature di `index{...}` in un file di LaTeX, ottenendo un file pulito senza marcatura di indicizzazione: questo può essere utile nel caso si voglia indicizzare di nuovo o con criteri diversi il documento. Il programma è scritto in Snobol4, usando Zenity come interfaccia grafica, il che lo rende portabile quasi al 100%. Per Windows si fornisce anche un file eseguibile autonomo.

1 Problem statement

Within the given scope, the aim of this program is very simple, and it is not meant to substitute human activity in this field. Complex concepts cannot be indexed by a simple program like this one, which at the moment is able to deal with single words only. However, lexical lists such as those found in humanistic manuscripts[1] can be effectively prepared using the present program. Database indexing is a different issue, of course, and retrieval and indexing is a crucial occurrence for any researcher specializing Internet philosophy and technology. Indexing of texts written in natural languages (as opposite to database, for example) is a form of text representation that can be very important for the future of a given text, e.g. its influence in research. If a text features a well structured index, potential readers will be able to locate bits of information they are interested in without being obliged to go through the whole book. To quote the words of a specialist, Marie-Francine Moens, «current text representations are often restricted to only certain terms that frequently occur in the text, or to all words from the beginning of the text, or to sentences that contain frequent terms. We assume that a representation that reflects the content in a semantically rich way will help solving the information retrieval problem in future systems» (Moens, 2002, 228). More recently, De Keyser's book on indexing features a very clear chapter defending automatic indexing (De Keyser, 2012).

Moens admits that making a suitable index often needs that some terms are manually indexed, especially when a term must be "distilled" from a concept, requiring an abstraction capability (Moens, 2002, 225), and I do agree: the procedure hereby proposed requires manual activity before and probably also after running the program. Many commercial programs on the market offer automated indexing or machine-aided indexing; the present program, obviously limited to LaTeX files, carries out a task similar to what is offered by a popular indexing program.[2]

In conclusion, although automated indexing *cannot be anathemised* as twenty years ago,[3] I still believe that computer-aided indexing, as opposite to automated indexing, is likely to be a reasonable perspective. Not only the program at the present stage of development, but also all the items in the TODO list (see p. 66) are not automated but computer-aided functions.

1. See for example Milanese (2005).

2. I refer to `Sonar Bookends`, produced by Virginia Systems (`http://www.virginiasystems.com/`) and called «popular» by Browne e Jermey (2007, 186): «its claims are not extravagant: it can produce a concordance (a page number for every word); a list of proper nouns with page numbers; or an alphabetised list of words and phrases that you supply ('go' words) with the page numbers shown». Browne's book is rather old: however, I tested the demo of the Professional edition in August, 2016, and the program, priced $395 (the non-Professional series is cheaper) does what it promises to do. For an enthusiastic description of computer-aided methodologies see Stauber (2004, 300-303): the author refers to `Macrex`, a Windows program still in the market (see `www.macrex.com`).

3. See for example the accident of a program called `Indexicon`, as narrated by Mulvany (2005, 251-252).

Insofar as LaTeX is concerned, I have found two similar projects on CTAN. Martin Thorsen Ranange's program `intex` (see `https://www.ctan.org/tex-archive/support/intex`) is a very clever Python script, from some points of view similar to the present program. `intex` requires the user to add tags (`\co{...}`) in the LaTeX file, while `forindex` prepares the basic form of an index with no manual tagging. From my perspective, this is the most important difference.

Paul Isambert's package XElnd is similar to the program hereby proposed, but it is a XeLaTeX-only project (`https://www.ctan.org/pkg/xeindex`).

Two Italian contributions are worth mentioning. Claudio Beccari's *Uso del comando `\write18` per comporre l'indice analitico in modo sincrono* (Beccari, 2009) suggests a clever way to avoid «three or four asynchronous runs of `makeindex` and `latex` or `pdflatex`» using the `\write18` command; Cevolani's article «explains how to generate the index of names using `biblatex`» (Cevolani, 2010). Cevolani's procedure could be usefully integrated in the present program, while Beccari's instruction is used in the output produced by this program.

2 Program outline

2.1 General interface

The program interacts with the user through a graphical user interface. After the first window, displaying the name of the program and the current version (see fig. 1), the user is presented with a dialogue window, asking to choose between the two provided functions (see fig. 2). The second dialogue window asks for the name of the LaTeX file, as described below.

2.2 Function `doindex`

`doindex` reads a LaTeX file, using a list file, and enters index entries in the file according to it. Previously entered index entries are left unchanged, allowing for further indexing.

The input LaTeX file must have extension `tex`, in whatever case combination. If the user makes a wrong choice, the program outputs an error message and the same dialog window is displayed again. The user can interrupt the program at any time hitting the cancel button.

The list file is meant to contain all the words to be indexed. It must have the same name of the LaTeX file and extension `wls`; sub-entries are separated by tabs: up to 3 levels are possible, which seems reasonable for most documents[4]. Such a file can be easily prepared with any spreadsheet program, such as `Libreoffice calc`, `Excel`, and the like (or obviously with a text editor). See `test.wls` as example:

animals		
foreign words	French	ça
animals	cats	
animals	dogs	
foreign words	German	drücken
foreign words	Italian	évita
		MacIntyre
		Cicero
		food
languages	nouns	house
		Aristotle
		Maritain
sleeping@sleep		

No particular order in this file is required. Some users will prefer alphabetical order, others different arrangements: therefore, the program has no requirements concerning order/sort in this file. Entries as `sleeping@sleep` use the standard `makeindex` syntax and are left unchanged. The "logic" of this syntax is the same of `makeindex`, namely:

Class (level 1) – Class (level 2) – Item

In the previous table, some entries are placed in the 3rd column, the 1st and the 2nd being empty. We could imagine to fill the empty values as such, for example:

philosophers	Scottish	MacIntyre
philosophers	Roman	Cicero

However, writing "standalone" items in the first column is more natural, and therefore the program treats the following two fragments as equivalent:

		MacIntyre
		Cicero

MacIntyre		
Cicero		

Some people may prefer to organise their index thinking the other way round, for example:

		animals
ça	French	foreign words
	cats	animals
	dogs	animals
drücken	German	foreign words
évita	Italian	foreign words
MacIntyre		
Cicero		
food		
house	nouns	languages
Aristotle		
Maritain		
sleeping@sleep		

4. See De Keyser (2012, 13-14): he dislikes what he calls «indirect indexing», and proposes «to rely on good software that can 'explode' from a narrower term to a broader or vice versa when needed».

This taxonomy is opposite to the internal logic of `makeindex`[5]. For `doindex`, both systems are acceptable. It's a matter of personal choice: you can think either (1) or (2):

1. Plato belongs to the class "philosophers"
2. the class "philosophers" owns Plato

Plato ∈ Philosophers or Philosophers ∋ Plato is a matter of thinking FROM MEMBER TO SET or FROM SET TO MEMBER. The same applies to a possible intermediate class (such as PLATO – ANCIENT PHILOSOPHERS – PHILOSOPHERS). Just let the program know which system are you following, and be consistent, of course. See fig. 3.

The original LaTeX file is left unchanged. A new file is written, identified by `-ind`. For example, from `file.tex` you will get `file-ind.tex`. Of course, you'll have to run `makeindex` as usual, but Beccari's procedure is used in the file, reducing the need of manual runs.

Another useful function is called replacement. For example, in the text of a book it is normal to use surnames instead of full names: I may write

> Many modern philosophers – e.g. Annas and MacIntyre – notice that ancient ethics is a fascinating field of research.

but in the index of names I will not refer simply to *MacIntyre* but to *MacIntyre, Alasdair*[6]. A replacement file lists the simple form – the form that `doindex` would use without further instruction – and the full form. The two values will be separated by tabs:

```
MacIntyre	MacIntyre, Alasdair
Cicero	Cicero, Marcus Tullius
Annas	Annas, Julia
```

Naturally you could use this also for other substitutions, such as geographical names:

```
Monaco	Monaco, Principauté de
Westminster	Westminster, City of
```

The replacement list is optional (see fig. 4). There are no restrictions on filenames and extensions: this is because a scholar may wish to have several lists of replacements and use them for all of his works, not only for a particular publication. A dialogue window asks the user to select a replacement list, if any—if the selected file does not contain a replacement list, the program fails.

5. DE KEYSER (2012, 13) calls this indexing organisation «inverted index terms» – confusing the class / member relationship with full names (to refer to his examples, «Art, Babylonian» and «Joyce, James»). The present program distinguishes among the two different structures.
6. For the general problem see MOENS (2002, 88).

2.3 The function `cleanindex`

The function `cleanindex` removes \index{...} sequences from a LaTeX file. The program can be used e.g. if a user is not happy with the indexing of a file and wants to start it all over again.

The input LaTeX file must have extension `tex`, in whatever case combination. If the user makes a wrong choice, the program outputs an error message and the same dialog window is displayed again. The user can interrupt the program at any time hitting the cancel button.

The original file is left unchanged. A new file is written, identified by `-noind`. For example, from `file.tex` you will get `file-noind.tex`. In this file, lines concerning `makeindex` are left but commented, in order to avoid an empty *Contents* section in the output. You can uncomment the lines as soon as you want to reindex the file.

3 Installation

3.1 GNU/Linux and other *nix systems

Unzip the archive in a directory of your choice. A new directory, called `Forindex`, will be created. Run `forindex` with your favourite program launcher or from terminal. If you prefer to compile `snobol4` for your particular system, download the sources from `http://www.snobol4.org/csnobol4/curr/`; compilation is straightforward (generally just `./configure`, `make`, `make install`), but do read the `README` file. The most recent version of the interpreter is 2.0 (2015).

3.2 Windows

The package offers an `exe` file compiled with Spitbol (see (`http://www.snobol4.com`). Make a directory (a subfolder of `Programs`, for example) and copy all the files there.

3.3 Macintosh

Still untested. The main problem is `zenity`: a new project looks promising (`https://www.macports.org/ports.php?by=library&substr=zenity`).

4 Test files

Please test the program on template files provided herein, named `atest.tex` and `atest.wls`, `btest.tex` and `btest.wls`. `atest.wls` uses the *Item – Class 1 – Class 2* logic, while `btest` follows the *Class 2 – Class 1 – Item* logic. The output files will be called `atest-ind.tex` `btest-ind.tex` using `doindex`, `atest-noind.tex` or `btest-noind.tex` using `cleanindex`.

5 Bugs and TODO

The program supports Unicode chars limited to Latin extended alphabet and Greek (complete). *Unicode files only*: since other encodings are rapidly

disappearing, adapting the program to latin1 or similar encodings is not a priority.

List of features that I would like to add[7]:

1. Index also included files.
2. Introduction of optional styles, such as italics for the most important locations of a word.
3. Support for several indexes (e.g. names, places, and general).
4. Option to generate a rough index of all the words (to produce a preliminary version of the `wls` file).
5. Support to index words listed with regular expressions. E.g. `read*` should index *read*, *reads*, *reading*, *readings*, all under the same heading *read*.
6. Optional separator for the `wls` file, e.g. comma.
7. Add Unicode support. % partially done, 1.0

6 Acknowledgements

6.1 Version 0.1, 2005 (slightly adapted)

The program `ixgen` gave me the idea of `forindex`. Many thanks to Oscar Lopez for this very good program[8].

Some questions sent by Carlo Pellegrino (Università di Modena e Reggio Emilia, Italy) gave me the idea of transforming a very rudimentary script into a general purpose utility. Maurizio Loreti (Università di Padova, Italy) sent me very useful remarks on the problems of automatical generations of indexes, which I made use of in the introduction to this text.

My warmest thanks to Phil Budne (phil@ultimate.com) for making his excellent CSNOBOL interpreter available. Many thanks to the community of Snobol users, particularly to the members of the list snobol4@mercury.dsu.edu, and, among them, to Gordon Peterson (https://www.linkedin.com/in/gordonpeterson) and to Rafal M. Sulejman (rafal@engelsinfo.de) whose `vim` syntax files are a daily blessing.

6.2 Addendum for version 1.0, 2016

I would like to add a word of gratitude to the members of the Italian TEX usergroup, GUIT (http://www.guitex.org) and particularly to Ivan Valbusa (Verona) for his generous help.

7. This is the list of desired features of version 0.1, published in 2005. Unicode, at least, is (partially) tagged as «done». Much more to do.

8. `ixgen` appears to be now (2016) a commercial program for `Framemaker`. I do not know what has happened to the LaTeX program with the same name. See http://www.fsatools.com/index.htm.

7 Screenshots

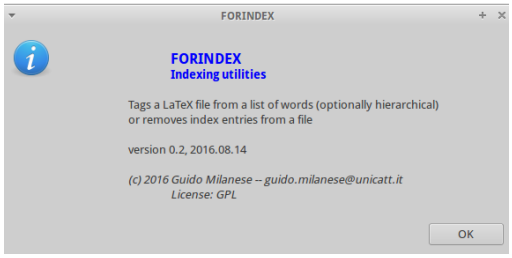

Figure 1: Initial message

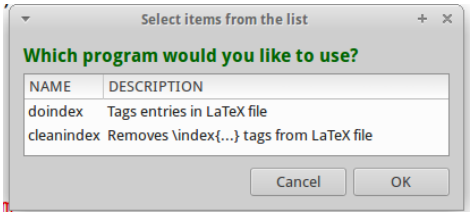

Figure 2: Choice of function

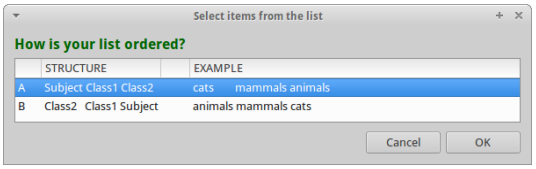

Figure 3: Order of `wls` file

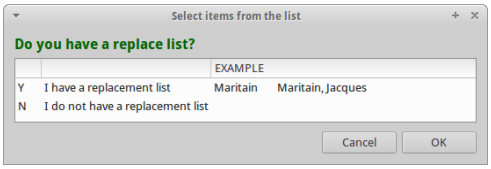

Figure 4: Optional replacement list

References

Beccari, C. (2009). «Uso del comando `\write18` per comporre l'indice analitico in modo sincrono». *ArsTEXnica*, (8), pp. 76–78. URL http://www.guitex.org/home/numero-8.

Browne, G. e Jermey, J. (2007). *The indexing companion*. Cambridge University Press, Cambridge. URL http://www.loc.gov/catdir/toc/ecip0620/2006028722.html.

Cevolani, G. (2010). «Indice dei nomi automatico con biblatex». *ArsTEXnica*, (9), pp. 31–38. URL http://www.guitex.org/home/numero-9.

De Keyser, P. (2012). *Indexing: from thesauri to the semantic web*. Chandos information professional series. Chandos, Oxford – Cambridge – New Delhi.

Milanese, G. (2005). *Censimento dei manoscritti noniani*. Pubblicazioni del Darficlet, N.S. 225. DARFICLET "F. Della Corte", Genova.

Moens, M.-F. (2002). *Automatic Indexing and Abstracting of Document Texts*. The Information Retrieval Series. Kluwer Academic Publishers, Boston, MA. URL http://dx.doi.org/10.1007/b116177.

Mulvany, N. C. (2005). *Indexing books*. Chicago guides to writing, editing, and publishing. University of Chicago Press, Chicago – London, 2ª edizione. URL http://www.loc.gov/catdir/toc/ecip057/2005004214.html.

Stauber, D. M. (2004). *Facing the text: content and structure in book indexing*. Cedar Row, Eugene, OR.

▷ Guido Milanese
Università Cattolica del Sacro Cuore, Milano–Brescia
guido dot milanese at unicatt dot it