

Use of the `\write18` feature for composing indexes

Claudio Beccari*

1 Introduction

Any \LaTeX user who has composed an index knows the trouble one gets in when s/he has spent a lot of time collecting the index entries, with the correct formatting, and eventually finishes his/her paper but forgets the last three runs in order to have the correct final index typeset at the end of the paper. The three runs are in order:

1. Run \LaTeX or $\text{pdf}\LaTeX$ with the command `\makeindex` in the preamble; this will collect the index entries, their formatting, and the page numbers where the entries are located in the document; this information gets written down in a file with extension `.idx` that contains all the entries with the related information in the order they were collected.
2. Run the program `makeindex`, possibly with a personalized style file and specific options in order to produce the sorted entry list with their ranking and page numbers. This information gets written into a file with extension `.ind`; this file contains the whole `theindex` environment with each entry marked with an `\item` or `\subitem` or `\subsubitem` prefix, together with the relevant formatting instructions for both the entries and the page numbers.
3. Run again \LaTeX or $\text{pdf}\LaTeX$ to process the final index file, possibly by inputting it by means of a command such as `\input{\filename}.ind`.

These three steps can be executed in just one run if you make use of the command `\write18{\textit}}`.

Modern \TeX distributions contain an executable file and a format file that have been created with special switches in such a manner that the interpreter `pdftex` may produce either a `.dvi` or a `.pdf` output file; it may make use of the commands of the extended `etex` interpreter, or refrain from doing so; it may or may not “shell out” to the system and ask it to perform some actions before resuming its main typesetting task.

Here I will exploit this last feature in order to produce the actual index in one run of $(\text{pdf})\LaTeX$.

*This is the translation of the original paper, published in this issue, titled *Uso del comando `\write18` per comporre l'indice analitico in modo sincrono*. The translation will be published in TUGboat, and appears here courtesy of both the author and TUGboat.

2 Auxiliary files

When the command `\makeindex` is executed while the preamble gets processed, a new output stream is opened in order to write the index information into the `.idx` auxiliary file; at the same time, that command activates the various `\index` commands within the document that actually write the information into the auxiliary file. The name of this output stream is assigned to the internal \LaTeX kernel control sequence `\@indexfile`.¹

This file has the specified extension `.idx` and its name is the same as that of the (main) `.tex` file that is being typeset; the name of this file is kept into the \TeX control sequence `\jobname`, that may be used by any suitable macro.

Since the index is typically the last thing to be typeset in a document, `\input{\filename}.ind` is likely the last command before the `\end{document}` statement.

So, in order to sort and produce `\jobname.ind` from `\jobname.idx`, we have to do essentially two simple operations:

1. Close the output stream identified by the internal macro `\@indexfile`, and
2. execute the `makeindex` command in order to produce the new updated `.ind` file.

3 The commands

We have to execute “immediately” the following two commands:

```
\immediate\closeout\@indexfile
\immediate\write18{
  makeindex -s style.ist \jobname.idx}
```

just before the `\input{\jobname.ind}` statement or its equivalent. Of course, if no specific index style file `style.ist` is being used, the whole clause `-s style.ist` is omitted.

The `\immediate` \TeX primitive instructs the interpreter not to delay the execution of the subsequent command; without it, the \TeX commands concerned with input and output streams, including the actual read or write commands, are

1. This is what the \LaTeX kernel and the standard classes do. Some nonstandard classes may use other control sequences to hold the information about the index output stream: for example the class `memoir` by default uses the control sequence `\jobname` extended with the suffix `@idxfile`, but, since `memoir` may build up several indices, each one has its own output stream with a different name, that gets specified with the optional argument of the command `\makeindex`.

deferred until a page is shipped out. Obviously `\closeout` closes an output stream, in our case the one specified by the subsequent macro `\@indexfile`.

The `\write18{<text>}` command writes onto the special output stream numbered “18”; this stream is equivalent to writing the `<text>` as a system command line in the command window of the Windows family of operating systems, of an xterm window of the Unix family of operating systems, or of the terminal window of the MacOSX operating system. When the system command finishes its processing, control returns to the TEX interpreter that goes on from the very point it had suspended its processing in order to have the operating system perform the required task.

The presence of the `@` character in the output index stream name ‘`\@indexfile`’ requires that a `\makeatletter` command be issued before the first statement and a `\makeatother` be issued just after the first statement, thus changing and reverting its category code.

On the other hand, if those two statements are defined with a macro contained in a personal `.sty` file, there is no need to change the category code of `@`. This would also be a good occasion to add useful bells and whistles to the new command, for example, checking if the output stream really exists and may be closed; checking that `\jobname.idx` exists in order to convert it; provide for an optional argument to specify the index style file so as to insert the style file clause in the system call; and warning and error messages could be added so as to let the author know if anything is inconsistent. This is not the proper place to add long stretches of TEX or LATEX code, but any reader should be capable of customizing a new command suitable for his/her needs.

4 The results

By issuing those commands, or the new command defined in the personal `.sty` file, just before giving the command `\input{\jobname.ind}` (or equivalent), the index turns out to contain exactly the index entries and page numbers that were collected during that current (pdf)LATEX run, and there is no need to remember to update the index file after any correction to the source file(s), since the entries and the page numbers are always up to date.

I find these simple macros extremely valuable, and I use them myself in all my (recent) documents, after I realized the possible benefits deriving from the `\write18` command. This facility is not widely used, because it makes it possible for TEX to execute external applications exposing the typesetter to the risk of executing malicious code; therefore some TEX distributions are distributed with this facility disabled by default.

Nevertheless there is some good news in this regard: TEX Live 2009, and a subsequent release of MiKTEX, will contain a restricted version of the `\write18` command, such that `\write18` will be enabled by default, but only a restricted list of “safe” programs (listed in a configuration file) will be allowed to be executed. And, happily, `makeindex` is one of those safe programs.

I imagine there are many other possible applications of the `\write18` facility: for example, the format transformation of graphic files is already available in the package `epstopdf.sty`; it’s up to the users to discover new ones.

▷ Claudio Beccari
claudio dot beccari at gmail
dot com