BLUe's Index

The principle and some more

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
`cgl@rc.service.rug.nl`

Abstract

The creation of a modest index within a one-pass TEX job is proposed. In general a proof run and a final run are needed. The markup for the index entries is the same as used by Knuth for *The TEXbook*. The process is controlled by the tags: `\sortindex` and `\pasteupindex`. The file which emerges after `\sortindex` can be enriched by hand, for example to include 'see also' and the like. Sorting keys can be specified too, as well as items which have to be ignored for the sorting. The macros have been developed for use with English documents. To leave open the use with other languages the ordering has been parameterized in an ordering table.

Keywords: Compatible extension, education, index, macro writing, number ranges, one-pass job, ordering table, plain TEX, reusable software parts, software engineering, sort keys.

1 Introduction

Making an index is an art. The fundamental problem is

What to include in the index?

Computer-assisted indexing is not simple either. Issues are

- the markup of keywords or phrases
- to associate page numbers
- to sort and compress raw Index Reminders (IRs), and
- to typeset the result.

The latter opens the Pandora box of markup and layout, which prompts for a two-pass job.

Up till now the sorting is done outside of TEX.[1] However, producing a modest index in a one-pass TEX job is possible. In 'Sorting in BLUe,' I already touched upon the issue of sorting IRs.

I'll build upon manmac's[2]

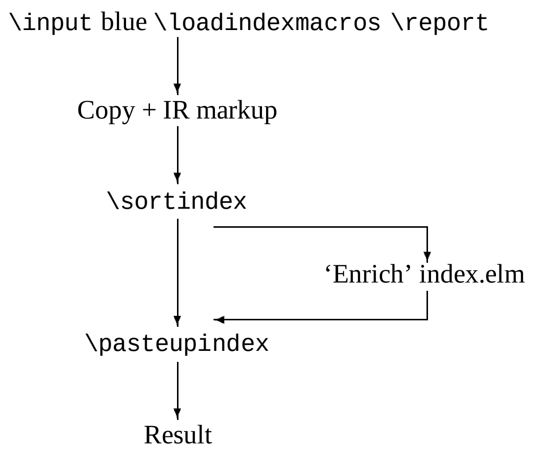
- syntax and types of Index Reminders (IRs)
- writing to a file
- associating page numbers, and
- the formatting in two-columns.

2 The process and files involved

Manmac stores the raw IRs in the file index. I read the file index[3] into an array for internal sorting. After sorting I reduce the entries[4] and write the result to the file index.srt. Then, I transform index.srt into the file index.elm.[5] The result is typeset via `\pasteupindex`.

2.1 Schematically

It comes down to



[1]The index for the TEXbook was done by a multi-pass job. In proofmode the raw IRs are written to the file index, and in the final run the enriched file of index entries is encluded in the script, preceded by special markup definitions.

[2]In contrast with Knuth I refrained from the full stop of each entry in the typeset index.

[3]Default index is the value of the toks variable `\irfile`, which is used in `\sortindex`.

[4]Those which differ by page number are collected into one entry.

[5]Default index.elm is the value of the toks variable `\indexfile`, which is used in `\pasteupindex`. The transformation abandons the IR syntax. The part which specifies the kind of IR is deleted and the word part marked up accordingly.

3 Why?

For a professional index I agree with Knuth that one can better sort and otherwise enrich the index outside of TeX. This obeys the separation of concerns principle. However, there is nothing against it to provide macros for producing modest indexes completely within TeX. Objections[6] will faint away hopefully, because of the possibility to *enrich the sorted and compressed index (file) outside of TeX*. The latter is a step beyond manmac. There the raw IRs are written to the file index, while here the sorted and compressed IRs can be worked upon. My work is a compatible extension of manmac, meaning that one can also start from the raw IRs, if one wishes.

A side-effect is that the power of TeX for practical problems is demonstrated.

4 Disclaimer

The macros are not foolproof, :-). Don't use as part of the IR

- TeX's special symbols
- symbols with special catcodes, for example active symbols like the circumflex ^, or the tilde ~
- as part of the IR a control sequence which is not accounted for.[7]

5 Notations and definitions

`manmac.tex` stands for the macros used by Knuth for formatting his Computers and Typesetting series of books, especially the TeXbook and the METAFONT book. Index reminders denote the markup of script words, to be included in the index. IRs is an abbreviation for Index Reminders. `\ea`, `\nx` denote `\expandafter` and `\noexpand`. OTR stands for output routine. FIFO stands for First In First Out, as described in 'FIFO and LIFO sing the BLUes.' SiB denotes my 'Sorting in BLUe' work.

6 Example of use: From the TeXbook

Let us take for this example some IRs from the first chapter of the TeXbook. I took only part of it and introduced `\newpage` now and then to enforce pages with zero, one or more IRs. I used `blue.tex` (with manmac embedded).

```
\input blue.tex \loadindexmacros
%Text from first chapter TeX book
Hence the name \TeX, which is an
uppercase form of $\tau\epsilon\chi$.
^^{TeX, meaning of}%modified
^^|\tau|^^|\epsilon|^^|\chi|
\newpage%no IRs next
Insiders pronounce the $\chi$ of \TeX\ as
a  Greek chi, not as an ‘x’, so that
\TeX\ rhymes with the word blecchhh.
\newpage%three IRs next
In fact, ^{TEX} (pronounced {\sl tecks\/})
is the admirable {\sl Text EXecutive\/}
processor developed by^{Honeywell HIS}.
Since these two system names are
^^{Bemer, Robert}
pronounced quite differently, they should
also be spelled differently.
\newpage%one IR next
The correct way to refer to \TeX\ in a
computer file, or when using some other medium
that doesn’t allow lowering of the ‘E’,
is to type ‘^|TeX|’. Then there will be no
confusion with similar names, and people will
be primed to pronounce everything properly.
\sortindex\pasteupindex
\bye
```

which yields as raw IRs (in file index)

```
TeX, meaning of !0 1.
tau !2 1.
epsilon !2 1.
chi !2 1.
TEX !0 3.
Honeywell HIS !0 3.
Bemer, Robert !0 3.
TeX !1 4.
```

and as index (in file index.elm)

	Bemer, Robert 3	`\tau` 1
	`\chi` 1	TEX 3
	`\epsilon` 1	`TeX` 4
Index	Honeywell HIS 3	TeX, meaning of 1

7 Example of use: Accents and the like

I'll show how to mark up Knuth's four types of IRs and how to mark up accents, font switching, and spaces as part of the IR. The last IR markup shows that the representation of page numbers as a range comes out automatically too.

```
\input blue.tex \loadindexmacros
Types of IR
 0  ^{return}
 1  ^|verbatim|
 2  ^|\controlsequence|
 3  ^\<syntactic quantity>

Accents in IR markup  ^{\’el\‘eve!},
font changing in IR markup ^{\bf bold}
and spaces in IR markup ^{control\ symbol}

Control sequences ^{\TeX, and \AmSTeX}
^{Lamport and \LaTeX}

brackets ^{\tt< \rm and \tt>}
\newpage range representation ^{return}
\newpage                      ^{return}
\sortindex\pasteupindex
\bye
```

The above yields in file index as raw IRs

```
return !0 1.
verbatim !1 1.
controlsequence !2 1.
syntactic quantity !3 1.
```

[6] If any :-).

[7] In the section Looking forward I'll explain how a user can allow for control sequences as part of the IR markup.

```
\'el\`eve! !0 1.
\bf{}bold !0 1.
control\ symbol !0 1.
\TeX, and \AmSTeX{} !0 1.
Lamport and \LaTeX{} !0 1.
\tt{}< \rm{}and \tt{}> !0 1.
return !0 2.
return !0 3.
```

and in file index.elm as index

< and > 1	Lamport and LaTeX 1
bold 1	TeX, and AMS-TeX 1
control symbol 1	return 1–3
`\controlsequence` 1	⟨syntactic quantity⟩ 1
Index élève! 1	`verbatim` 1

8 Index Reminders

IRs are at the heart of the process. Knuth distinguished 4 types to facilitate the outside processing. I'll adopt his IRs syntax and types.

8.1 Syntax

Knuth's IRs obey the syntax[8]

⟨*word(s)*⟩ !⟨*digit*⟩ ⟨*page number*⟩.

The digits 0, 1, 2, or 3 denote the types: words, verbatim words, control sequence, and syntactic quantity. A user does not have to bother about the digits, nor about the page numbers. Knuth has adopted the accompanying conventions for the word(s) of IRs.[9]

Mark up	Typeset in copy*	IR
`^{...}`	...	`... !0` ⟨*page no*⟩.
`^|...|`	`|...|`	`... !1` ⟨*page no*⟩.
`^|\...|`	`|\...|`	`... !2` ⟨*page no*⟩.
`^\<...>`	⟨...⟩**	`... !3` ⟨*page no*⟩.

* |...| denotes manmac's, TUGboat's,... verbatim.
** in `\rm`.

For the user the word(s) are important. The allowed markup for the IRs and the result in the copy are given in the accompanying table.

8.2 Markup

The markup for IRs is near to natural. Precede the entry by a circumflex (or two circumflex(es) in case of a silent index entry).[10]

Example *(IR markup.)*

```
^{text, e.g. \'el\`eve!}
^|verbatim text|
^|\controlsequence|
^\<a metalinguistic variable>
%and for silent ones, double the ^
^^\<a metalinguistic variable>
%from the TeX book
{\sl^{ligatures}}
|'$|^|\,||$''|
^^{markup commands, see control sequences}
%from Looking forward section
^{Lamport and \LaTeX}
```

Spaces

are as always difficult. In the IR they separate parts of the IR, and are used in the word part.

Just typing a space has as effect that it will be neglected during sorting.

The markup '\␣', a control space, will yield a space subject to sorting, according to the ordering table.

`\space` markup will be neglected during sorting. This token is default member of the set of to be ignored control sequences. It will be set in the index as \␣.

Example *(Spaces)*

The following markup

```
\input blue.tex \loadindexmacros
Spaces test
^{\space}%an ignored cs
^{a\ a}  %control space
^{aa}
^{a\ b}
^{a \TeX}
^{a\ \bf a}
^{\TeX book}
^{xyz beta}%space neglected in sorting
^{xyza}
^|\space|
\sortindex\pasteupindex
\bye
```

yields as file index

```
\space {} !0 1.
a\ a{} !0 1.
aa{} !0 1.
a\ b{} !0 1.
a \TeX {} !0 1.
a\ \bf a{} !0 1.
\TeX book{} !0 1.
xyz beta{} !0 1.
xyza{} !0 1.
space{} !2 1.
```

and as file `index.srt`

```
\space {} !0 1.
a\ \bf a{} !0 1.
```

[8] In contrast with my previous given syntax it seems that Knuth was less restrictive. Earlier I overlooked that the entries are ended by a period.

[9] See *The TeXbook* 424 for the IR types, and what is typeset in the result. In `\vref` the markup is inserted as replacement text of `\next`. What is set in the index is governed by the macros which are included after `\begindoublecolumns` in the *The TeXbook* script.

[10] Silent IRs mean that these will appear only in the index, not on the page.

```
a\ a{} !0 1.
a\ b{} !0 1.
aa{} !0 1.
a \TeX {} !0 1.
space{} !2 1.
\TeX book{} !0 1.
xyza{} !0 1.
xyz beta{} !0 1.
```

Explanation. `\space` belongs to the set of to be ignored control sequences, ICSs for short. This means that it is skipped with respect to sorting, except when it occurs as the last token of the word part. In that case they are ordered as a space, that is according to the lowest value. This explains the position of '`\space`.'

'`\TeX`,' and '`\TeX` book,' are subject to the default sort keys.

'xyza' precedes 'xyz beta,' because the space is silent. When word ordering is preferred a \␣, a control space, must be included.

8.3 Writing the IRs in index

comes with manmac. The writing is done in two phases: first while processing the script, and second in the OTR where the page numbers are attached. The (manmac) macro which does the first part of the writing is

```
\def\makexref{\ifproofmode
   \bgroup\def\ {\string\ }%
   \xdef\writeit{\write\inx{\text{}
    !\xreftype\space
    \nx\number\pageno.}}\writeit
   \egroup
   \else\ifhmode\kern0pt\fi\fi
   \ifsilent\ignorespaces\else\next\fi}
```

I don't write in the margin. In Appendix C the full-BLUe macro has been provided. Font changing control sequences, accents and `\space` are written as a 'string' in the file index. Actually, I introduced also the sorting on sorting keys. Because of the various uses of `\space` I introduced `\spaceseparator`.

9 Sort and compress: `\sortindex`

The functionality of `\sortindex` is to transform the file of raw index reminders — index — into the file with sorted and compressed index entries — index.elm.

All we have to do is to

- write the raw IRs in the array, and keep the upper bound of the array in `\n`,
- sort with the right comparison macro (`\cmpir`, and at the lower level for the word part `\nxtwindex`), next to the use of the ordering table `\otindex`
- reduce the index entries by collecting the same entries which differ by page number
- write the result to the file index.srt, and transform this into index.elm.

```
\def\sortindex{%Nov 1994, cgl
%Purpose:
%To sort IR file.
%Input: default index is sorted
%      (file specified in \irfile)
%Output: file index.elm.
\newpage\immediate\closeout\inx
\filetoarray{\the\irfile}
\immediate\write16{Sorting n=\the\n.
    Please wait, O(nlog n) process.}
\let\cmp\cmpir\let\nxtw\nxtwindex
\otindex\let\ \space
\sort
{\let\spaceseparator\space
 \setupnxtokens\def\ {\nx\ }%
 \immediate\write16{Range reduction.}
 \redrngtofile{index.srt}
 \immediate\write16{After reduction and
  writing to file index.srt; n=\the\n.}
 \immediate\write16{Transform
      index.srt-->index.elm.}
 \tawfiletofile{index.srt}{\the
                 \indexfile}}}
```

with auxiliary `\def\setupnxtokens{%`

```
 \def\process##1{\def##1{\nx##1}}%
  \ea\fifo\the\conseqs\ofif
 \def\process##1{\def##1{\string##1}}%
  \ea\fifo\the\consyms\ofif
}
```

The sorting within TeX has been treated in SiB, and will not be repeated in this article. A user must supply a comparison macro.

In Appendix C the full-BLUe version has been incorporated.

9.1 Storing in an array

The storing of data from a file into an array has been treated in SiB. A slightly modified version of the macro reads

```
\def\filetoarray#1{%#1 is file name
 \immediate\openin\inxin=#1\relax
 \ifeof\inxin\immediate\write16{File #1 empty
   or non-existent.}%
 \fi \n\kzero\continuetrue
 \loop\ifeof\inxin\continuefalse\fi
 \ifcontinue\advance\n1 \immediate
 \read\inxin t\ea o\csname\the\n\endcsname
 \repeat\advance\n-1
 \immediate\closein\inxin}
```

9.2 Comparison

For sorting I make use of my macros as released in SiB. Comparison needs a multiple key. This means that at the outer level we have to supply `\cmpir`[11] and at the lower level we have to compare words, and digits. The words are handled by `\nxtwindex` and the numbers are compared via an `\ifnum`. The comparison macro for IRs reads

```
\def\cmpir#1#2{%#1, #2 defs
%Result: \status= 0, 1, 2 if
%        \val{#1} =, >, < \val{#2}
\ea\ea\ea\decom\ea#1#2}
```

The crucial macro for comparison of the word part of the IR reads

```
\def\nxtwindex#1#2{%
```

[11]Mnemonics compare index reminder.

```
%Function:
%On input: #1 contains the ‘word’
%As result: #2 contains the value
%   of the first non-ignored token as given
%   in the ordering table.
%#1 contains the rest of the ‘word’
 \def\pop##1##2\pop{%
 \gdef#1{##2}\def\pophead{##1}}%head and tail
 \ea\pop#1\pop%split in head and tail
 \ignores\ea\ea\ea{\ea\the\ea\conseqs
                       \the\consyms}%
 \ea\loc\pophead{\the\ignores}%
 \iffound\ifx\empty#1 \chardef#2=0
         \else\nxtwindex#1#2\fi
 \else
 \ea\let\ea#2\csname ot\pophead\endcsname\fi
}
%with toks variables
\conseq{\c\space\bf\it\rm\tt\TeX\sub}
\consyms{\‘\’\"\ˆ\˜}
```

The idea is to process the ‘word’ argument by argument.[12] However, some tokens are not relevant for the ordering and have to be ignored. This is done by collecting all tokens to be ignored in the toks variable `\ignores`, and compare `\pophead` with this string. The above can be extended to control sequences to be sorted on separately provided sorting keys. See the Looking forward section.

For `\decom` and other low level macros related to sorting within TEX see ‘BLUe's Format,’ or ‘Sorting in BLUe.’

In Appendix C the full-BLUe versions have been incorporated.

Ordering

A fundamental issue with indexes is the ordering. The ASCII table is not suited because lowercase and uppercase letters differ by 32. I decided to rank these as equal, more precisely to assign the lowercase ASCII values to both. I prefer from the following the left column above the right one

```
el                            el
Elève                         em
em                            Elève
```

Moreover, accented letters are not part of ASCII. How should we order for example e, é, è, ê, ë? I decided to rank accented letters equal to those without an accent, because I prefer from the following the left column above the right one

```
el                            el
élève                         em
em                            élève
```

I know that non-letters precede letters but what about there relative ordering? I decided to stay as close as possible to the ASCII ordering.

Then there is the problem of digits. In IRs they come as part of the word(s) and as page numbers. For the latter I used the numerical ordering. For the former I used the alphabetical ordering.[13]

Furthermore, a user can select the so-called ‘word ordering,’[14] by \␣, TEXnically a control space, as markup for a space. Personally, I like from the following the first column better than the second

```
sea lion                      seal
seal                          sea lion
```

Ordering table

This ordering ‘table’ is simpler than the one released in SiB, because accents have been ignored. Furthermore, I adhere mostly to the ASCII ordering, as can be seen easily.

```
\def\otindex{%Parameters: Ordering ‘table’
%Special cases
\ea\chardef\csname ot \endcsname=0
\ea\chardef\csname ot\space\endcsname=0
%{|}˜ come in ASCII after lowercase
%ˆ is active character
%Bulk according to ASCII
\def\process##1{\ea\chardef
   \csname ot##1\endcsname=‘##1 }
%lowercase letters
\fifo abcdefghijklmnopqrstuvwxyz\ofif
\chardef\otij=‘y \chardef\otIJ=‘y
%other characters}
\fifo!"##$&’()*+,-./0123456789:;<=>?@
      []_‘\ofif
%assign lowercase values to uppercase letters
\def\process##1{\ea\chardef
  \csname ot##1\endcsname=\lccode‘##1 }
\uppercase{\fifo
  abcdefghijklmnopqrstuvwxyz\ofif}
}
```

Attention needs TEX's specials, in particular the escape character \, the circumflex ˆ, and the percent %.

9.3 To be ignored tokens

In practice I needed things like `\tt` as part of the IR, which must be neglected while sorting.[15] I decided to ignore those tokens while sorting and to include the tokens in the final index.elm as such.

Another realistic approach is to add this kind of markup later in the file index.elm.

9.4 Reduction of entries

The functionality is that the IRs which differ by page number are collected into one entry with the page numbers represented efficiently, for example in ranges. The macros from SiB have the functionality

```
\def\redrngtofile#1{%Reduction of \1...\n,
%with range representation of page numbers.
%The result is written to file #1.
...
}
```

[12]Not token by token, beware!

[13]I could have applied a look ahead mechanism and use numerical ordering throughout. Maybe another time.

[14]This means that spaces precedes all letters. A space as such is neglected in the ordering.

[15]The reason is that `<, and >` are used and then printed wrongly.

```
%
\def\prcrng#1{%Prints the numbers so far if
%the new number differs more than 1 from the
%last. If the difference is 1 the range is
%extended.
...}
%
\def\strnrs{%Accumulates numbers in \nrsrng.
%either as a range or as such.
%\frst stands for first number
%and \last for last number. If they equal the
%number is stored, if they differ by 1 both
%the numbers are stored separated by \sepn,
%and if they differ by more than 1
%\frst--\last is stored.
...}
```

For the macros see Appendix C.

9.5 Transformation index.srt→index.elm

When we inspect the copy for the index in the TEXbook file then we'll find that the entries of the enriched file don't obey the syntax of the IRs. The part with !$\langle digit\rangle$ has disappeared. Pondering about this made me agree with Knuth, as usual.[16] It is no longer functional! The file can better be considered as copy as such, to be processed in the final run. Because of this I transformed index.srt, the file of sorted and compressed IRs, into the file index.elm, with the coding !$\langle digit\rangle$ absorbed as markup in the word part.

```
\def\tawfiletofile#1#2{% #1 from file
 \continuetrue        % #2 to file
 \immediate\openin\inxin=#1\relax
 \immediate\openout\inx=#2\relax
 \loop\read\inxin to\IR
   \ifeof\inxin\continuefalse\fi
 \ifcontinue\trfandwrite\IR
 \repeat
 \immediate\closein\inxin
 \immediate\closeout\inx
}
%with auxiliaries
\newread\inxin\newwrite\inx\newtoks\indword
\def\splitintoks#1 #2 #3.{\indword{#1}%
  \chardef\digit=#2\relax\def\pagenrs{#3}}
%
\def\trfandwrite#1{\ea\splitintoks#1%
 \immediate\write\inx{\nx\noindent
  \ifcase\digit{\the\indword}\or
   {\nx\tt\the\indword}\or
   {\nx\tt\char92\hbox{\the\indword}}\or
   $\nx\langle\hbox{\the\indword}\nx\rangle
    $\fi\spaceseparator{\nx
   {\oldstyle\pagenrs}}}
```

Explanation

In `\trfandwrite` I used the toks variable for storing the word part, because when writing it comes out handy that `\the` is a one-step expansion. Note that `\trfandwrite` takes care of the markup for `\pasteupindex`. The `\noindent` is inserted to enforce horizontal mode, especially in presence of accents at the beginning of the word.

In `\tawfiletofile` the test for the end of file looks peculiar.[17]

10 Typeset: `\pasteupindex`

In the *The TEXbook* the typesetting of the enriched index entries is treated on 261–264. Let us start from there and distill our specifications.

The typesetting of main and subsidiary entries is discussed given the file of index entries. This is intertwinned with the page break mechanism in relation to what should appear in the running headlines.

My specifications for typesetting the index are

- represent the four IR types the same as in the TEXbook
- set in two-columns, balanced, possibly preceded by one-column copy
- set subsidiary entries analogous to the TEXbook
- indent continuation lines by 2em
- indent subsidiary entries by 1em
- underline page numbers which represent the definition or the main source of information
- represent a page number in italics when that page contains an instructive example of the concept in question.

Essentially nothing new. The challenge is how to implement this, such that an index can be handled in a one-pass job.

In order not to make things too complex, I'll postpone the handling of subsidiary entries until the Looking forward section. Let us say that this is a feature for the 'next release,' of `blue.tex`. The enrichements such as representation of numbers in italics or underlined, which have all to do with the wish to direct readers to the main source or to instructive examples, are left to the manual editing of the index.elm file. The reason is — In agreement with Knuth? — that this is difficult to foresee at the time when the IR markup is inserted in the script. Moreover, it complicates the sorting et cetera process.

In the mean time users can edit index.elm — read add markup — and provide the necessary macros in for example `\preindex`. In short follow Knuth.

The compressed and sorted aray of index entries is set via the invocation `\pasteupindex`.[18] The contents of `\preindex` and `\postindex` are used appropriately.

```
\def\pasteupindex{%Nov 1994, cgl
%Purpose:
%To set index in (balanced) doublecolumn.
%The index is preceded by contents of
%\preindex and followed by contents of
%\postindex.
%Input: default index.elm is set
%       (file specified in \indexfile).
```

[16] I noticed furthermore, that some IRs did not make it into the final index. Apparently, given the other IRs, he decided to delete less relevant ones.

[17] Remember that TEX appends a `\par` to the input file.

[18] As modification for the pair `\begindoublecolumns` and `\enddoublecolumns`, because the latter is too much intertwinned with manmac. For an explanation of the underlying macro the reader is referred to the *The TEXbook* 416–417.

```
%Biased by manmac's \begindoublecolumns
 \newpage\begingroup
 \def\space{{\tt\char32 }}%
 \the\preindex\par
 \pageheight\vsize
 \pagewidth\pagewd%anachronism
 \parindent1em
 \output={\global\setbox\partialpage=
  \vbox{\unvbox255\bigskip}}%
 \eject
 \output={\bluedoublecolumnout}%
 \hsize=8.5cm \vsize=51cm%blue.tex values
 \parskip0pt plus.8pt\relax
 \obeylines\everypar{%
    \hangindent2\parindent}%
 \let\par\endgraf
 \let\sub\endgraf
%
 \input\the\indexfile\relax
%
%endpasteupindex part biased by
%manmac's \enddoublecolumns
 \output={\balancecolumns}\eject
 \endgroup
 \pagegoal=\vsize\the\postindex
}
%
%auxiliaries adapted from manmac
\def\bluedoublecolumnout{%
%Biased by manmac's doublecolumnout
 \splittopskip=\topskip
 \splitmaxdepth=\maxdepth \dimen@=25cm
 \advance\dimen@ by-\ht\partialpage
 \setbox0=\vsplit255 to\dimen@
 \setbox2=\vsplit255 to\dimen@
 \blueonepageout\pagesofar
 \unvbox255 \penalty\outputpenalty}
%
\def\blueonepageout#1{%
%Biased by manmac's \onepageout
 \shipout\vbox{%
 \vbox to\baselineskip{\null
        \the\headline\vss}%
 \kern2ex
 \vbox to\pageheight{#1}%
 \kern1ex
 \the\footline
 }\advancepageno}
```

The file name is parameterized in a token variable `\indexfile`. Default is `\indexfile{index.elm}`. A user can specify his own file via

`\indexfile{`⟨*user index file*⟩`}`

10.1 Running headlines

The advanced mechanism of having the top entries and bottom entries appropriately represented in the headline has been treated in the TEXbook.

Looking at the indexes of the TEXbook and the more recent Graphbase book, I noticed that Knuth did not use it. In Appendix D he states that the index is not tall enough to justify the mark mechanism. It is really advanced and subtle, and for those who are interested, please peruse the TEXbook.

11 Customization

A user might want to interfere at the places

- to include other tokens to be ignored while sorting[19]
- to supply an ordering of his own
- to enrich the sorted and compressed file index.elm.

11.1 Adding tokens to be ignored

In general we don't know what control sequences stand for.

What are reasonable requirements to impose upon the handling of markup control sequences (cs for short)? In my opinion

- the cs must be defined
- `\makexref` writes the cs unexpanded
- ordering? unknown, and therefore neglected[20]
- `\setupnxtokens` guards that the cs-s are written, unexpanded, to index.srt and index.elm.

As a consequence I decided to neglect the 'in between' control sequences while sorting. For those who favour a one-pass job, I have provided the following, although it is simpler to add those control sequences to index.elm.

The extension of the set of to be ignored tokens can be done via

```
\def\add#1to#2{...}%See App C
%for example adding to set of control
%sequences
\add\hfil to\conseqs
%or control symbols
\add\'to\consyms
%Adding to the set of sort key pairs
\add\hfil{hfil}to\srtkeypairs
```

Each element from `\conseqs` is redefined such that the control sequence token is written to the file with a space appended.[21]

11.2 Modifying ordering

The most general way is to 'copy' the ordering table for modification.[22]

And what about a macro to add to the table?

This can be done easily, and superficially looks convenient for an innocent user. At the moment I don't trust the macros to be worthwhile for an innocent user, unless a very modest index has to be made. And this completes the circle: different ordering is not wanted, I guess.

11.3 Enriching the index

This use is necessary when for example

- control sequences have to be typeset
- special symbols are needed, or

[19]However, in the Looking forward section this has been generalized into the possibility to provide the control sequence with a sorting key.

[20]However, see the Looking forward section, ;-).

[21]`\noexpand` is used instead of `\string`.

[22]My `\fifo` is just a shortcut, which also prevents typos in assigning the ASCII values. For `\fifo`, see 'FIFO and LIFO sing the BLUes.'

- cross-references within the index are required.

The best way is to start from the index.elm file.

An example of use is that in the 4TEX manual the index is sorted on 4TeX, but in the index a different representation is desired. (The control sequence `\fourtex` has been used for typesetting instead of 4TeX.) For that purpose the file with IRs contains 4TeX and later this is substituted in the file index.elm by `\fourtex`.

Another approach is to supply pairs of control sequences and sorting keys. See the Looking forward section.

11.4 Typesetting the enriched file

When the default name is used — index.elm — just say `\pasteupindex`. For another file name assign this name to the toks variable `\indexfile`, prior to the invocation of `\pasteupindex`.

12 Tests

The macros have been tested on the file which was obtained from the TEXbook chapters 1 to 6. The (slightly adapted) file of raw IRs and the resulting sorted and compressed index entries have been included in the Appendices A and B. The test had to do with some 220 raw IRs.

A driver program, which starts from a file of IRs not necessarily generated by manmac, is

```
\input blue.tex
%\irfile{erik.15b}    %file with 6 entries
\irfile{erik.16b}     %file with 19 entries
%\irfile{indexdat.610}%file with 183 entries
%\irfile{erik.cgl}    %file with 228 entries
\sortindex\pasteupindex
\bye
```

Another test file was from the 4TEX manual. This makeindex file consisted of roughly 760 entries of the form `\indexentry{...}{<number>}`. I first transformed this file — by TEX — into a file obeying the IR syntax, via

```
\newwrite\erik
\immediate\openout\erik=erik.cgl
\def\indexentry#1#2{%
 \immediate\write\erik{#1 !0 #2.}}
```

Then I edited the file in order to
- remove the backslash(es) in the middle of the word part
- adjust IR type for control sequences (0 into 2) such that the sorting et cetera went smooth.

As result I obtained 343 sorted and compressed index entries. For typesetting some back substitutions had to be made
- in between backslashes inserted again
- the substitution/insertion of special control sequences.

The total sorting time on my Mac Classic II was roughly 20 minutes, with 12200 IR comparisons, and 54149 words of memory used.[23] As format I used blue.tex.[24]

The user's guide for BLue's Format system takes some 400 IRs. The processing of the index takes 1 hour on my Mac, and a little over 1000 save stack positions.

A final test was about copy with font changes and accents as part of the IRs. This file has been supplied in the second example of this article.

13 Robustness

The weak point in this automated complicated process is the specification of the IRs. When wrong ones are supplied low-level TEX error messages will emerge, and definitely will put off a user. My only remedy to this is
- don't hardly use markup within the IR
- don't use active charcters as part of the IR
- don't use TEX special characters, especially backslashes apart from a single control sequence (a type 2 IR.)

I also print in the log file the number of elements to be sorted: n. This must be greater than zero. When n=0 is printed, the file of raw index entries is apparently empty, and 9 out of the 10 cases the asynchronous behaviour of the OTR is the cause, meaning that the file was already closed before it was written.

When the file is empty, as argument of `\filetoarray`, a message is delivered in the log file.

In the macros I have left some `\immediate\write`s, preceded by %, which can be used to follow what is compared. This is handy when spotting an out of order IR.

14 Looking backward

It all started with sorting an array in SiB. In the early stage of this paper I adhered the model to allow writing IRs to a file and also, as option, to write directly to the array. Because of the latter, I had to modify the OTR such that the array elements were extended with page numbers. I used the mark mechanism and it worked.[25] I abandonned this option for simplicity reasons. When restricted to writing to a file a reader does not have to bother about the OTR, and no redefinitions of the array elements are needed.

A price I had to pay for flexibility is the generation of the replacement texts for control sequences to be ignored, each time in every invocation of `\makexref`.
Another 'inefficiency' is the 'generation' of the ordering table.
Perhaps, I should just have included the 'tables' as such instead.
A final point is the general conflict with control sequences already in use.

[23] A dummy job which just stores the IRs but does not sort nor reduce the number of entries needed 34884 words of memory.

[24] While proofing this work, it needs less hand work, because the control sequences can be accounted for via sortkey pairs. See the Looking forward section.

[25] However, there was still a detail to be fixed: the first mark was also written in the main vertical list.

15 Looking forward

Is it realistic to expect that there will be another user besides me? I don't think so, because history has it that manmac as such has been neglected at large, and I don't see why people would nevertheless prefer my work, which is so intimately connected to manmac.
Whatever the future has in store, let us go on.

15.1 Subsidiary entries

What are the requirements for this? Let us assume that the markup for a subsidiary entry starts with `\sub`. IMHO the following specs are relevant

- `\sub` should be neglected in the copy
- write `\sub` as string to the file index
- neglect `\sub` while sorting
- 'reduce' entries which differ in subentries by suppressing the main entry except for the first time, while typesetting the index element
- indent the subsidiary entry by `\parindent` while typesetting.

The above specs can be realized as follows

- `\let\sub=\relax`
- include `\sub` in `\conseqs`
- the reduce problem is similar to the typesetting of references, where the author part is printed once for different publications. I found that in $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX, and used it in 'BLUe's References.'
- add `\let\sub=\endgraf` in `\pasteupreferences`.

Actually, the above has been incorporated in blue.tex.

15.2 Sorting keys

So far we have mostly neglected tokens while sorting. But, . . . is it do-able to sort each control sequence according to a sorting key?

This comes down to

- provide a way to specify for sorting key pairs, that is pairs of a control sequence and its sorting key
- sort on the sorting key
- set the index entries with the embedded control sequences according to their definitions.

Example *(Use of sorting keys)*

Suppose that we have

```
\add\fourtex{4tex}to\srtkeypairs
\add\fourtex to\srtkeys
%or \setupsrtkeys
Copy with ^{IR \fourtex}
%
\sortindex %with 4tex for \fourtex
\pasteindex%Set ‘IR \fourtex{} <pagenumbers>’
\bye
```

then the file index will contain the IR

```
IR \fourtex{} !0 ⟨pageno⟩
```

The above index element will be sorted on 4TEX. The key issue in the implementation is to extend `\nxtwindex` by the test `\pophead` $\in$ `\srtkeys`. If the test yields true sort further on `<sorting key><tail>`, that is insert the sorting key. Actually the above has been included in blue.tex with as defaults

```
\conseqs{\c\space\bf\it\rm\tt\sub\relax}
\consyms{\`\'\"\^\~}
\srtkeypairs{\TeX{tex}
             \LaTeX{latex}
             \AmSTeX{amstex}}
\srtkeys{\TeX\LaTeX\AmSTeX}
```

In order to use this nice feature extend the script as follows

```
\add...to\srtkeypairs%one or more pairs
\setupsrtkeys        %extends the set of
                     %sort keys
...%copy proper
\sortindex
\pasteupindex
```

The suppression of the word of the main entry with multiple sub entries can be implemented too. For the moment I refrained and wait for user response.

16 Availability

The macros are part of BLUe's Format System, and stored in tools.dat. The system is released on NTG's 4(All)TEX CD-ROM and the CTANs.

Used from manmac are the IR macros, the circumflex ^ and its auxiliaries, next to some macros for handling the doublecolumns, for example `\balancecolumns`. Some macros from SiB are used too, especially the sorting macros.

17 Acknowledgements

Erik Frambach thank you for the file of 4TEX, your assitance, and your sound judgement. As usual Jos Winnink helped me again to procrust the markup of the article into `maps.sty`.

18 Conclusion

The macros work smoothly for a modest and practical index. I have used the macros for the user's guide of BLUe's Format system 'Publishing with TEX.' The macros serve an educational purpose, and stimulate thinking.

My added value to manmac is, that a modest index can be processed in a one-pass job. Accents, to be ignored tokens, and sorting keys can be supplied. More complex indexes can be processed via a proof run, editing of index.elm, and a final run with index.elm inserted as copy.

For a foolproof approach it is necessary to look ahead for general tokens. The problem is that we can't account for all the control sequences or active characters to be used. For indexes of modest complexity my limited approach is good enough. The bonus is simplicity throughout.

In general it is difficult to know when to stop, as Schumacher in his precious 'Small is beautiful' already pointed out. My case rest.

References

I borrowed ideas undoubtedly from all the material I read. Most of the literature I read is contained in my literature database lit.dat. To my knowledge no-one handled the preparation of an index completely within TEX, before.