Experimenting with makeindex and Unicode, and deriving kameindex

Antoine Bossard, Keiichi Kaneko

Abstract

When writing and editing a long document such as a book or thesis, it is almost inevitable to include an index. The purpose of this part of a document is to collect several keywords that appear in the main text, listing them together with the corresponding pages numbers, in order to facilitate information search inside the document. Document typesetting is often realised with the LaTeX system, especially for professional and academic editing. The LaTeX typesetting system embeds via the *makeindex* subsystem an indexing feature, which semi-automatically generates a document's index. However, makeindex has not been developed to support Unicode and thus fails in most cases at generating indices that include multilingual entries, and especially entries of different writing systems. In this paper, we first review the, even scarce, multilingual capabilities of makeindex, before deriving *kameindex*, a solution to these internationalization issues. Importantly, kameindex replaces one component of the makeindex subsystem, thus retaining overall compatibility with makeindex. After illustrating the results achieved by kameindex, comparison with related works is conducted.

Sommario

Quando scriviamo o editiamo un documento lungo come un libro o una tesi, è quasi inevitabile includere un indice. Lo scopo di questa parte di documento è elencare una serie di parole chiave che appaiono nel testo corredandole con i corrispondenti numeri di pagina, così da facilitare la ricerca dell'informazione nel documento. La composizione di documenti avviene spesso in LaTeX, specialmente per pubblicazioni professionali e accademiche. Il sistema di composizione LaTeX incorpora uno strumento di indicizzazione grazie a *makeindex*, che genera semiautomaticamente l'inidce del documento. Purtroppo makeindex non è stato sviluppato per supportare Unicode e quindi nella maggior parte dei casi sbaglia a generare gli indici che includono voci multilingue, e specialmente voci in diversi alfabeti. In questo articolo, inizialmente recensiamo le pur scarse capacità multilingua di makeindex, prima di derivare *kameindex*, una soluzione a questi problemi di internazionalizzazione. Cosa importante, kameindex sostituisce un componente di makeindex, mantenendo quindi la compatibilità generale con makeindex. Dopo aver illustrato i risultati permessi da kameindex, faremo una comparazione con altri lavori correlati.

1 Introduction

Indexing is about gathering the keywords of a document (e.g., thesis, book) and listing them together with the corresponding page numbers in a special section – the index – usually included at the end of the document. Indices are especially useful for long documents: without such information, it would be very tedious to search inside the document. Indeed, paper documents do not provide an automated search feature as with electronic ones. The usage of the document typesetting system LaTeX is widely spread, especially throughout the academic community. Thanks to the *makeindex* subsystem, LaTeX is capable of semi-automatically generating indices. This process is semi-automatic since the writer has to manually declare index entries throughout the document.

An important and well-known shortcoming of makeindex is its lack of Unicode support. Effectively, makeindex has been developed to process ASCII files only, that is, makeindex works on a byte per byte basis. Consequently, makeindex is in most cases failing at realising multilingual indices – especially when mixing writing systems. In this paper, the first objective is to review the, even lacking, multilingual capabilities of makeindex. The second objective is to derive from the exposed makeindex issues a Unicode-capable indexing solution, *kameindex*, that will replace one component of the makeindex subsystem, thus importantly retaining compatibility with the original makeindex system.

This research is part of a wider project that aims at improving the support of the Japanese writing system by computers and information and communication technology (ICT) in general. In a previous work, we have described an unrestricted character encoding for Japanese (Bossard and Kaneko, 2018). Further advancing this subject, we consider in this paper an application – indexing in LaTeX – where Japanese support is lacking, and propose concrete solutions to address this issue. The long term objective is to support the previously proposed character encoding with this indexing application.

A large part of this paper is addressing multilingual documents. We rely on Unicode and its UTF-8

implementation (Unicode Technical Committee, 2011) for character representation (encoding) purposes. LaTeX source code mentioned hereinafter is compiled with XeLaTeX (Robertson and Hosny, 2017). Therefore, font loading is easily realised with the `fontspec` package and, for instance, its `\newfontfamily` command.

2 Reviewing makeindex capabilities

The fundamental features of makeindex are first briefly presented. Then, workarounds for multilingual (Unicode) documents are presented.

2.1 Basic features

The makeindex utility and corresponding LaTeX package (Lamport, 1987) are used to automatically generate indices from a set of index entries declared inside the source document (call the `\makeindex` command to activate indexing). An index entry is declared with the command `\index`. makeindex is then in charge of compiling the index by separating entries (per the first letter), sorting them, merging identical ones, collecting and merging page numbers, calculating page ranges and so on. Finally, the output of the resulting index is requested to the compiler with the `\printindex` command.

The `\index` command can also be used to declare sub-entries: `\index{entry!subentry}`, with at most two levels of sub-entries. The text of an entry can be customised: `\index{entry@\textit{entry}}`, that is, the left part of the `@` symbol is the index entry itself, used for instance for index sorting, while the right part is the text to be actually printed for the entry. Page numbers can also be formatted: `\index{entry|textit}`. Cross-referencing is also available: `\index{entry|see {other}}` and `\index{entry|seealso {other}}`. Page ranges are declared with `\index{entry|(}` and `\index{entry|)}`.

2.2 Workarounds for internationalization

Even though makeindex does not support Unicode, it can be tricked into rather properly handling Unicode index entries as shown below.

First, consider languages of the Latin alphabet but with accented letters. French is used as example here. Because makeindex does not support Unicode, it will consider bytes (i.e., ASCII characters) one by one when collecting entries and for sorting. This is problematic here since accented letters are out of sequence with other letters: in UTF-8, the three letters 'd', 'e', 'f' are each encoded with one byte, 100, 101 and 102, respectively. The accented letter 'é', which is sorted in the lexicographical order just as 'e', is encoded with two bytes: 195 and 169. Hence, from a makeindex point of view, the index

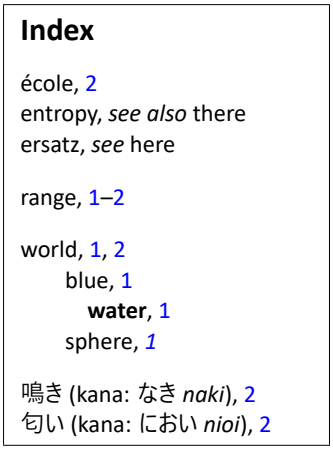


Figure 1: Sample index generated by makeindex and printed in the resulting PDF. Multilingual index entries are correctly separated and sorted, except for non-ASCII ones.

entry "école" will be sorted after "double", "entre" and "froid" where it should be "double", "école", "entre" and "froid".

This problem can be solved by using the `@` feature of index entries to provide supporting text for makeindex processing (e.g., entry sorting). Concretely, instead of declaring the entry `\index{école}`, declare `\index{ecole@école}`. In other words, accented letters are "de-accented" for index entry declaration.

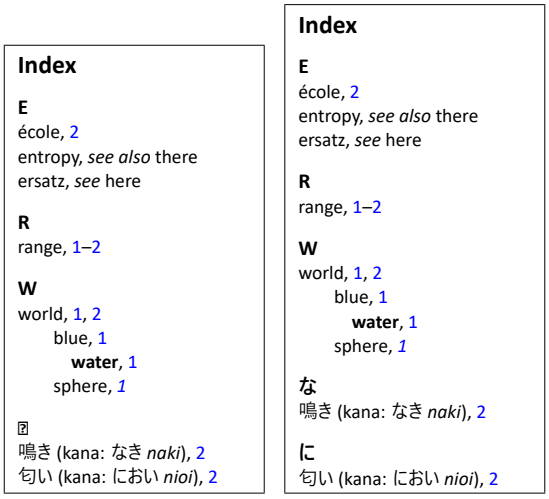
Next, consider Japanese. Most characters used in Japanese are Chinese ones. These characters are only partially ordered in the Unicode standard. Moreover, this is (partial) semantic ordering, which is completely unrelated to Japanese lexicographical ordering. The latter, which is formally defined in the JIS X 4061 standard (Japanese Industrial Standards Committee, 1996), relies on the reading of a character. Such pronunciation information is represented by the small Japanese character subset kana; these characters convey no semantic information, only reading information. Fortunately (of course it was purposely done so), kana characters are represented in Unicode in their lexicographical order. For instance, the consecutive three kana characters な, に and ぬ (*na*, *ni* and *nu*, respectively) are encoded with the byte sequences 227 129 170, 227 129 171 and 227 129 172, respectively. Hence, it makes no difficulty for makeindex to sort kana index entries. Therefore, by reusing the same trick as with accented letters, it is possible to have Japanese index entries: `\index{いす@椅子}`, where the word 椅子—"chair", read *isu*—is represented by the corresponding kana characters いす.

In addition, just as with accented letters, kana characters can have variants: for instance, ば *ba* and ぱ *pa* are two variants of は *ha*. In order to correctly sort entries, such variants (here ば and ぱ) need to be replaced by the "normal" character (here は). Other variants include small characters:

for instance, ゆ for ゆ, both *yu*. More details can be found in Japanese Industrial Standards Committee (1996).

It should be noted here that the default document font is unlikely to support Japanese. Hence, after loading a new font with the `fontspec` package, say `\fntjap`, the index entry would be declared as `\index{いす@{\fntjap 椅子}}` in order to properly print the index. The index key (i.e., the left side of `@`) is only used for index entry declaration (not printing) and thus does not require the selection of an appropriate font.

The features and workarounds described in this section are illustrated in Figure 1. It can be noted that index entries are correctly separated into groups according to entries' first letters, with as a notable exception the non-ASCII entries which are improperly grouped: the Japanese entries なき and におい start with a distinct glyph and should thus be classified into two distinct groups; they are not.



(a) (b)

Figure 2: (a) Illustrating the makeindex issues with Unicode index entries: entry grouping and group heading problems. (b) These problems are solved by kameindex.

3 Advanced features: deriving kameindex

In addition to the briefly presented features, makeindex also enables the usage of styles for typesetting the index (Chen and Harrison, 1988; Chen, 1991). In practice, the user declares style directives inside a style file. Explicitly printing letter groups (i.e., group headings) is probably the most used style feature; it is activated with the style directive `headings_flag 1`. Such a style file is then set as second input of makeindex, in addition to the index file (the index generation flow is described below).

As mentioned earlier, when using multilingual index entries such as those of Figure 1, as makeindex is processing bytes (ASCII characters) rather than (Unicode) characters, it fails at correctly classifying entries. This issue is even more problematic when index entry groups are materialised by group headings; see Figure 2a. Again, besides the invalid group heading, the two Japanese entries should be in distinct groups.

Hence, even though workarounds exist (see Section 2), Unicode support is severely restricted with makeindex. Nonetheless, we next show that encountered issues do not prohibit completely the use of makeindex when processing Unicode documents. Index entry classification, group headings and such Unicode-related issues are addressed by kameindex. This program is used in place of the makeindex program for the index file generation process as shown in Figure 3. This figure details the document (.pdf) generation flow, starting from a LaTeX source file (.tex). The LaTeX program generates an index file (.idx) and a PDF file. (The index generation process indeed requires two passes.) Together with a style file (.ist) if any, the index file is then streamed into the makeindex (replaced by kameindex for Unicode support) program, which generates a distinct index file (.ind) for processing, beside the source file (.tex), by the LaTeX program. One should note that LaTeX is here a generic appellation which should be replaced by XeLaTeX since dealing with Unicode.

The results obtained from makeindex and kameindex for the same input can be compared in Figures 2a and 2b, respectively. It can be noted that kameindex correctly handles Unicode index entries, while still correctly handling the other ones. One important issue is font selection for non-ASCII group headings, since the group heading glyph may not be present in the default font, as with Japanese headings. In kameindex, we have implemented automatic font selection: the font selected is the one used for the first index entry of the group (or the default font if no font is specified). This is a relevant choice since the entry label and its sorting information are in the same language (unless a peculiar sorting, one that does not reflect label sorting, is specified by the user – a case non addressed in this work). In addition, as headings are often emboldened (i.e., the command `\textbf` is applied to the heading), one should not forget to declare a bold font when loading a font that does not include a bold variation for glyphs. This can be done for instance with the font loading command `\newfontfamily\fntyu{YuGothR.ttc}[BoldFont=YuGothB.ttc]` for the Microsoft Windows Japanese font "Yu Gothic". Manually specifying a bold font is usually unnecessary when loading a font by font name rather than by file name. Last but not least, one should note that, as in this font loading example, the font command name (i.e., `\fntyu`) is prefixed with `\fnt`; this is required by kameindex for font command detection.

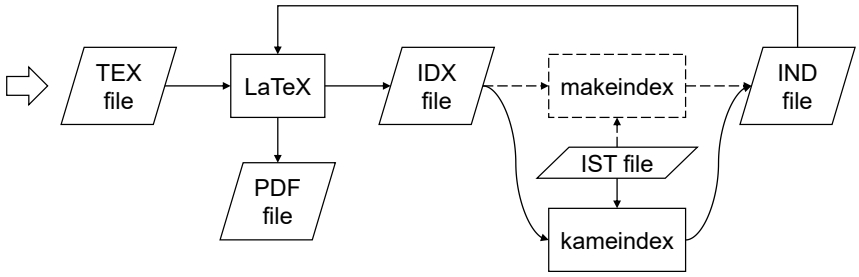


Figure 3: Index generation flow with kameindex instead of makeindex.

Additional non-trivial features of kameindex include: ranges, page numbers and ranges merging (e.g., a same entry containing the page numbers 1, 4, 6 and the page ranges 2–3, 5–7 will be output as the single range 1–7); subentries; cross-references; label, page number and group heading formatting; `hyperref` (Rahtz and Oberdiek, 2017) support (see Figure 2: page numbers are coloured to indicate links).

4 About Chinese and Korean

Chinese words are conventionally ordered alphabetically according to their pinyin transliteration (romanization). Hence, the workarounds mentioned in Section 2.2 are applicable for the indexing of Chinese words with makeindex. Concretely, index entries are declared in pinyin (non-accented) but with the entry text in Chinese characters. Also, since group headings will consist of Latin letters, they should be typeset with the default font. Thus, the font command for Chinese should not be prefixed with `fnt` so as to avoid using the Chinese font for group headings (kameindex detects font commands prefixed with `\fnt` as explained previously). For example, the index entry `\index{hanyu@{\chn 汉语}}` for the word 汉语 *hànyǔ* "Chinese [language]"; the font command here is `\chn`.

The case of Korean is more complex. The two main writing systems of Korean are hanja and hangul, with the former being based on Chinese characters. Thus, indexing of hanja entries can be done as with Chinese: romanization is used for entry sorting and grouping. As for hangul, it involves a special set of characters, with a particular ordering. First, it is important to note that hangul entries are sorted correctly by makeindex (and kameindex) since hangul glyph Unicode code points are ordered in the appropriate order. The problem which thus remains for indexing is grouping and group heading generation.

Precisely, a hangul word always starts with a consonant, called the "initial". Hence, grouping and group headings for Hangul words are necessarily based on one of the hangul initials (e.g., 'ㄱ', 'ㄴ' and 'ㄷ'). Therefore, it suffices to declare hangul index entries with the first glyph's initial as prefix. For example, `\index{ㅎ한글@{\fntkor 한글}}` for the word 한글 *hangeul* whose first glyph's (i.e., 한) initial is ㅎ. It is indeed enough to prefix the entry with the first initial, which will be used as group heading. Effectively, the entry being initially sorted correctly, adding as prefix the first initial does not impact the ordering. It is like prefixing the entries "abc", "acd" and "bde" by duplicating the first letter: "aabc", "aacd" and "bbde" retain the original entries' order.

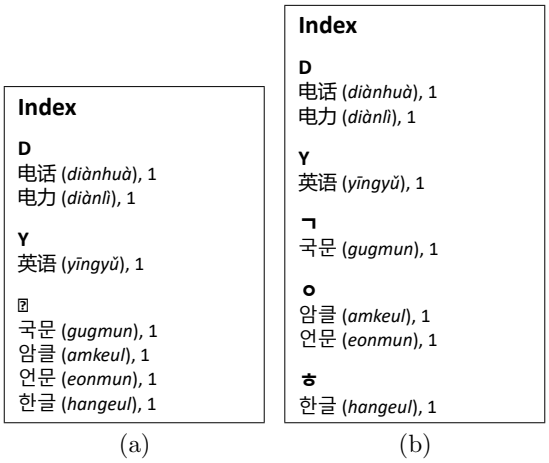


Figure 4: Illustrating Chinese and Korean index entries: (a) makeindex output, (b) kameindex output.

While makeindex would fail at generating properly an index for Korean entries, a sample kameindex output in the case of Chinese and Korean is given in Figure 4.

5 A more advanced example

We give in this section a more complete index example, using different alphabets. In addition to the Latin alphabet, possibly with accented letters as previously seen, Russian index entries, that use the Cyrillic alphabet, are included. When using Cyrillic letters, makeindex fails at grouping entries and creating group headers since characters are multibyte (i.e., not ASCII). Comparatively, kameindex works perfectly. As said for French, support text may be needed for Cyrillic "accented" letters as used for instance in Serbo-Croatian, Belorussian, and

Macedonian (e.g., 'Ѕ', 'Ѓ' and 'Ћ'). Greek (modern) entries, that use the Greek alphabet, are also included. Again, support text may be needed for Greek accented letters (e.g., 'ά', 'έ' and 'ή').

One should note that even though rendered similarly, some letters are distinct from a Unicode point of view. This is the case for instance with the Latin letter 'A' and the Cyrillic letter 'A'. Thus, index entries starting with such letters will be separated in two groups (i.e., Latin 'A' and Cyrillic 'A'). The same remark holds for example with the Latin letter 'E' and the Greek letter 'E' (i.e., uppercase 'ε').

An illustration of such a multilingual index is given in Figure 5; considering a same input, the makeindex and kameindex outputs are given in Figures 5a and 5b, respectively. Once again, one can see the shortcomings of makeindex. LaTeX commands for a few selected index entries are given in Table 1; it can be noticed that these are normal indexing commands, with nothing fancy. As mentioned previously, because Japanese characters are unlikely to be included in the default document font, the label (i.e., the right side of @) of the Japanese entry specifies the font `\fntyu`. Also, support text is provided to handle entries with accented letters and Japanese entries.

Table 1: LaTeX commands for selected index entries.

Command	Translation
`\index{юбиле́ йный}`	anniversary
`\index{εψιλον@έψιλον}`	epsilon
`\index{ecole@école}`	school
`\index{ㄱ국문@{\fntkor 국문}` `(\textit{gugmun})}`	national script
`\index{なき@{\fntyu 凪}` `({\fntyu なぎ} \textit{nagi})}`	calm (wind)

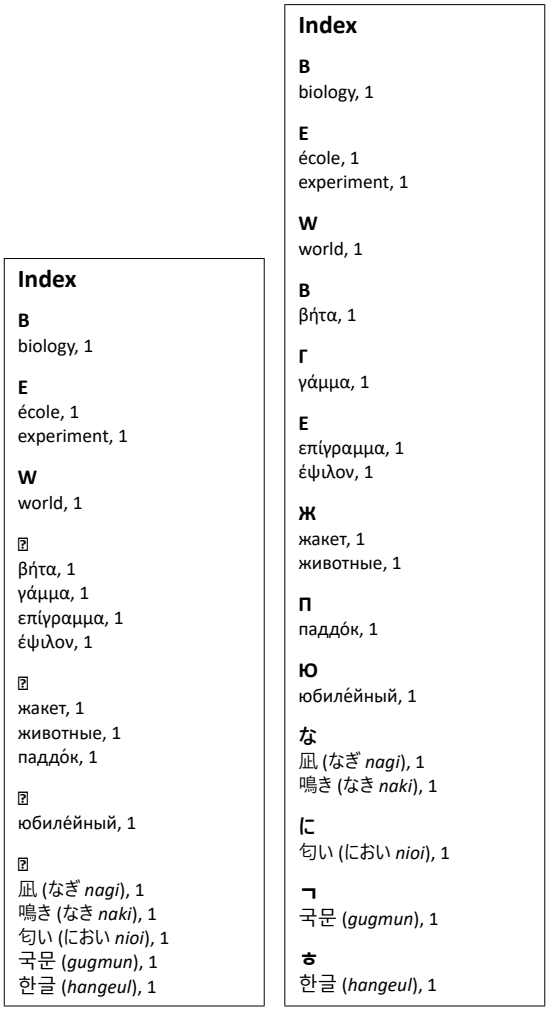


Figure 5: A more advanced multilingual index example: (a) makeindex output, (b) kameindex output.

6 Comparison and contribution

We have already shown how kameindex compares to, and supersedes, makeindex; see Section 3. In this section, we conduct additional comparison with the mainstream indexing system *texindy*, before summarising the contribution of this paper.

6.1 Comparison with related works

For multilingual documents, and Unicode in general, the indexing system *texindy* (Schrod, 2004) is recommended as a replacement of makeindex. Even though it is not our purpose with kameindex to replace *texindy*, it is worth noting the following three issues of *texindy*:

- *texindy*'s usage is comparatively complex;
- *texindy* is not compatible with `hyperref`;
- *texindy* requires manual language selection;
- *texindy* supports neither Japanese, nor Chinese, nor Korean.

All these issues are addressed by kameindex. First, the usage of kameindex is identical to that of makeindex and thus retains its usability; no additional complexity. Second, kameindex is fully compatible with `hyperref` as explained and shown in Figure 2. Third, no language selection is required with kameindex. Fourth, we have already illustrated the full Japanese support provided by kameindex.

6.2 Paper contribution

The contribution of this paper is summarised below. In addition to proposing kameindex as a Unicode-capable replacement for the makeindex program that generates index files (.ind), we have shown with concrete use cases the followings:

- When used wisely (see the discussion on workarounds in Section 2), makeindex retains a certain degree of usability with Unicode and multilingual documents. The remaining issues mostly concern the grouping of index entries,

including group heading generation, when entries are declared with non-ASCII characters.

- By replacing only a part of the makeindex subsystem (precisely, the makeindex program for index file (.ind) generation is replaced by kameindex), full Unicode support can be attained. The original makeindex package, providing for instance the `\makeindex` and `\printindex` commands, remains untouched (i.e., used as is).
- The index generation and rendering processes remain user-friendly: the only change is the call to kameindex in place of makeindex for index file (.ind) generation.

7 Conclusions

Indexing in LaTeX is conventionally realised with makeindex. However, makeindex has not been designed to support Unicode, which thus makes index creation for Unicode and multilingual documents difficult. Nevertheless, in this matter, makeindex is much more capable than often perceived. Effectively, as presented in this paper, even though designed without Unicode support, sensible usage can produce satisfactory results in a large panel of cases. The other cases are for instance the usage of index entry classification and group headers. We have proposed here the Unicode-capable kameindex program that is flow-compliant with makeindex. The usability of kameindex has been shown with concrete examples.

Regarding future works, supporting the previously described character encoding for Japanese (Bossard and Kaneko, 2018) is a meaningful objective. In order to avoid tempering with the XeLaTeX program, it would thus be necessary for kameindex to realise some character conversion when generating the index file (.ind). Also, completing the support of index style file (.ist) directives is one possible future work.

Acknowledgements

This research project is partly supported by The Telecommunications Advancement Foundation (Tokyo, Japan).

References

Bossard, A. and Kaneko, K. (2018). "Proposal of an unrestricted character encoding for Japanese". In *Proceedings of the 13th International Baltic Conference on Databases and Information Systems*. Springer, volume 838 of *Communications in Computer and Information Science*, pp. 189–201.

Chen, P. (1991). *makeindex(1): makeindex – a general purpose, formatter-independent index processor*. Unix man page. https://linux.die.net/man/1/makeindex (last accessed August 2018).

Chen, P. and Harrison, M. A. (1988). "Index preparation and processing". *Software: Practice and Experience*, **19** (9), pp. 897–915.

Japanese Industrial Standards Committee (1996). *JIS X 4061 "Collation of Japanese character string" (日本語文字列照合順番, in Japanese)*. Japanese Standards Association.

Lamport, L. (1987). *MakeIndex: an index processor for LaTeX*. Package documentation. https://ctan.org/pkg/makeindex (last accessed August 2018).

Rahtz, S. and Oberdiek, H. (2017). *Hypertext marks in LaTeX: a manual for hyperref*. Package documentation. https://ctan.org/pkg/hyperref (last accessed August 2018).

Robertson, W. and Hosny, K. (2017). *The XeLaTeX reference guide*. https://ctan.org/pkg/xetexref (last accessed August 2018).

Schrod, J. (2004). *texindy(1): texindy – create sorted and tagged index from raw LaTeX index*. Unix man page. http://xindy.sourceforge.net/doc/texindy-man.pdf (last accessed August 2018).

Unicode Technical Committee (2011). *The Unicode standard (version 6.0), §3.9 Unicode encoding forms D92, §3.10 Unicode encoding schemes D95*. The Unicode Consortium.

Appendix

The style directives for index generation that are supported by kameindex are listed in Table 2.

Table 2: The index style directives supported by kameindex.

Directive	Description
`headings_flag`	Activates headings if set to 1
`heading_prefix`	String used as heading prefix
`heading_suffix`	String used as heading suffix
`preamble`	String used as index prefix
`postamble`	String used as index suffix

The default values for the `preamble` and `postamble` directives are `''\\begin{theindex}\n''` and `''\n\n\\end{theindex}\n''`, respectively, purposefully matching those of makeindex. Besides, the two strings `''{\\bfseries\\large ''` and `''}\\nopagebreak\n''` are sample values for `heading_prefix` and `heading_suffix`, respectively.

▷ Antoine Bossard
Graduate School of Science
Kanagawa University
2946 Tsuchiya, Hiratsuka, Kanagawa
259-1293, Japan

▷ Keiichi Kaneko
Graduate School of Engineering
Tokyo University of Agriculture and
Technology
2-24-16 Nakacho, Koganei, Tokyo 184-
8588, Japan