TEXnical Tips for Producing a "Clean" Index

PUBLICATIONS TECHNICAL GROUP
AMERICAN MATHEMATICAL SOCIETY

This document has two main goals:

- to make you think about what kind of index(es) you want;
- to tell you what to do when something goes wrong.

To put this another way, we suggest how you should structure and fine tune your input so that you don't have to make repairs later.

Index terms are added to a document using the `\index` command; terms specified in this way are written out by the LaTeX job into an `.idx` file which is then processed separately by the makeindex program to create an `.ind` file which is, in turn, read back in the next time LaTeX is run to generate the printed index.

Only the barest outline of the basics for using the `\index` command and running makeindex are given here. More complete instructions are described in numerous sources (among them [La], [MG], and [Gr]), and you are referred there for details. However, for the most part, these references don't say what to do when the index doesn't turn out the way you expect. These "gotchas" are the main focus here.

The AMS document classes have not been tested with xindy, and compatibility is unknown. Likewise, no testing has been done with `utf8` character encoding, only with the original Knuthian method of encoding accented letters (`\'e`, etc.). Authors using AMS document classes should keep these limitations in mind.

Although this document is directed to AMS authors, the general principles apply to any LaTeX document with an index. If you have any suggestions for additions or improvements, please send them to mailto:tech-support@ams.org`tech-support@ams.org`.

Contents

Version Version 0.95, 14 December 2015.

1. The basics

To launch an index in a document, include the command `\makeindex` in the preamble and insert `\printindex` where you want the index to print (usually in the back matter). If you are using an AMS document class, do *not* include `\usepackage{makeidx}`; the necessary facilities for producing a single index are already built in.

A key decision for many books is whether to have only one index, or more than one, say for names, notation, terminology, etc. Instructions for preparing multiple indexes for one document are given in Section 7, along with some tactics for delaying such a decision while still entering complete index information in your file.

Most index terms are inserted in the source file as close as possible to the item being indexed; in order for an index to be useful, the cited page number(s) should be accurate, and if the commsnd to post an index entry is not right next to the term in the `.tex` file (e.g., it is separated by a lot of text or a paragraph break), the resulting page number may be incorrect. Input for a typical index entry will take this form:

```
... as defined by Lord Kelvin,\index{Kelvin, Lord} ...
This is called a link\index{link of knots} of knots.
```

Multi-level index entries

Up to three levels are permitted per entry, allowing multiple terms that begin with the same word(s) to be grouped. These are entered with the levels separated by an exclamation mark (`!`):

```
\index{centralizer}
\index{centralizer!in homomorphic image}
\index{centralizer!in ring}
```

which would appear thus in the index:

centralizer, 18–19
  in homomorphic image, 53
  in ring, 174

Influencing sort order

Sometimes makeindex will not sort an entry into the desired order, and it is necessary to provide a sort field. This is always true when the entry text begins with math coding or any command. The input begins with the sort field, followed by the at-sign (`@`), followed by the entry text coded as it should appear. A sort field should also be provided when a letter with a diacritic appears close to the beginning of the term. Some examples:

```
\index{C-module@$\mathcal{C}$-module}
\index{Miscellanea Analytica@\textit{Miscellanea Analytica}}
\index{p-group@$p$-group}
\index{Polya, G.@P\'olya, G.}
```

They will appear in the index (page numbers omitted here) as follows:

$\mathcal{C}$-module
Miscellanea Analytica
p-group
Pólya, G.

Sort fields may be included at any level, and are recommended, since a well-organized index makes it easier for readers to find what they are looking for.

Coverage of cross-references and sort fields will be expanded later.

Don't put comments inside an index entry

Index entries are copied verbatim into the file that will be used to produce the index. This includes a `%` sign and anything that follows it. When such an entry is read in with `\printindex`, the `%` will be interpreted as a comment, and everything following it will be ignored, often resulting in an error, but sometimes merely omitting the last part of the intended item in the index. If an error condition exists, it will not be detected by makeindex, but will fail only in the LaTeX run.

File usage and processing

Three types of files are involved in indexing:

- the source (⟨*filename*⟩`.tex`) file(s) with `\index{...}` entries;
- the file of raw index data (⟨*filename*⟩`.idx`, usually created by LaTeX);
- the file of sorted index terms (⟨*filename*⟩`.ind`, created from the `.idx` file by running makeindex) that will be input to produce the actual index.

Starting with the ⟨*filename*⟩`.tex` source files, processing to incorporate an index proceeds as follows:

- run LaTeX as many times as required to get rid of errors and create the `.idx` file;
- run makeindex ⟨*filename*⟩ to create the `.ind` file; this is a separate step *external to* LaTeX, and must usually be run from the command line;
- rerun LaTeX to generate the index.

The makeindex run will create a log file (⟨*filename*⟩`.ilg`). Resolve any errors or warnings by updating the affected entries in the source file, and check the index itself for possible problems. Make corrections as necessary (in the source file) and restart the procedure from the beginning.

LaTeX does not sort the index data; that is done separately by the makeindex program, which transforms the `.idx` file into an `.ind` file. This means that no matter how many times LaTeX is run, unless makeindex is also run, no index will be produced.

The final index should be prepared only after all textual corrections have been made and checked, and the page numbers are final. At that point, run makeindex on the "last" `.idx` file and rerun LaTeX twice more to incorporate the index and update the table of contents.

Always make changes in the source file

Although it's possible to edit an `.idx` or `.ind` file, it's a bad idea. Any change that requires LaTeX to be run again will create a new copy of the `.idx` file, destroying the old one. Even if there are no changes in indexed page numbers, if the files are used later for a new edition, corrections to the `.idx` or `.ind` file won't be in the source, and they will probably have been forgotten. Keeping the source updated and "clean" will ensure the most reliable results.

In order for the source file to be edited and corrected easily, it's a good idea to keep each index entry on a single line, inserting `%` signs as appropriate to avoid unwanted extra spaces, which in turn may result in separation of the index term from its target. (The percent sign should be attached to the term being indexed,

never placed *inside* an index entry. A comment inside an index entry will cause the entry to be truncated when the `.ind` file is read in, with confusing results; see page 11.) See below for tactics in applying this principle.

Differences between the AMS and basic LaTeX document classes

As mentioned earlier, the AMS document classes automatically incorporate the "standard" indexing facilities described in [La, Appendix A]; inclusion in the preamble of `\usepackage{makeidx}` will result in an error.

Cross-references within an index are also handled differently in the AMS document classes. The details are given in the next section.

2. Cross-references in an index

Two forms of cross-reference are common in indexes to mathematical books: *see* and *see also*. Basic LaTeX document classes provide only the first, with the command `\see`.

Owing to historical mischance, the AMS document classes cause command `\see` to result in *see also*. To sidestep this misfeature, the AMS classes provide `\seeonly` which will produce just *see*. (Unfortunately, the meaning of `\see` can't be changed, since it would cause problems with too many "legacy" books.)

These commands are entered as `|seeonly` or `|seealso` in an `\index` entry, as shown by these examples:

```
\index{Thomson, William|seeonly{Kelvin, Lord}}
\index{knots|seealso{link of knots}}
```

A vertical bar (`|`) replaces the backslash in the `\index` entry so that the command will withstand makeindex processing and be passed through intact to the next LaTeX run, in order to suppress printing of the page number for these entries.

The default *see also* entry will bs strung out at the end of the page numbers for the indexed item. However, it is often less confusing if the cross-reference is listed on a separate line at the end of the second-level entries. To accomplish this, a command `\indexalso` can be defined; place this code in the preamble of your file:

```
\providecommand{\indexalso}[3]{%
  \index{#1!zzzzz@\emph{\alsoname} {#2}}}
```

and input the entry as

```
\indexalso{knots}{link of knots}
```

It will appear in the index as

knots, ⟨*page numbers for real references*⟩
see also link of knots

An important feature of both *see* and *see also* entries is that they are not associated with specific page numbers. For this reason, and also so that they can be managed efficiently, all cross-reference entries should be input as a single block in your `.tex` file; a convenient location might be just before the `\printindex` command in the master file.

If more than one *see* or *see also* reference is needed for a single index term, include all of them in a single `\index` entry for that term, separated by semicolons.

3. Tactics for avoiding off-by-one page numbers

There are several situations in which page numbers in the index can be too low or too high by one:

- an index term referring to a chapter entered *before* the `\chapter{...}` line will always result in a page number that is too low by one (or even two, depending on where the previous chapter ended);
- a term referring to a section entered *before* `\section{...}` will get a page number too low by one if the section starts at the top of a new page;
- an index term referring to a theorem, definition, or similar element entered *before* the beginning of the environment will get a page number too low by one if the environment starts a new page; similarly, if the index term is entered *after* the end of the environment, the page number may be too high by one if the environment falls at the end of a page;
- an index term referring to a displayed equation entered *after* the end of the display will get a page number too high by one if the display ends a page;
- if an indexed term consists of more than one word and these words are split at a page break onto two pages, the page number assigned to such an entry may be either too low or too high by one depending on which word in the term the index entry is attached to.

Index terms referring to a new section or environment

An index term referring to a new chapter, section, or theorem-class object should always be entered as soon as possible *after* the heading in the file. Just after a `\label` (if present) is a good place.

Index terms referring to displayed material

An index term that refers to a display should be entered *before* the display rather than after it. There should be no blank line between the text and the following display; this will ensure that at least one line of text precedes a display on a new page, with the associated index term as part of the package.

If an index term is inserted within a display, a space will be added at the end of any control sequence within the index term, which may result in "duplicate" entries in the index if the same index term is used elsewhere. Thus it is best to input index terms only before displays. However, if a display consists of multiple lines, and a page break is necessary in the middle, an index term referring to material that may go onto the continuation page can be inserted within the display; a good place is just after the `\\` preceding the display line to which the term applies. Be sure to check the page numbers for such index entries carefully, and also check for unwanted "duplicates".

If placing an index term within a display is unavoidable, and duplicate entries result, identify those for which the extra space is added and precede the affected control sequence by `\string`, for example

```
\index{diffeomorphism group of $\string\Sigma$}
```

Index terms referring to names

The goal here is to choose the optimum reference word in the text to "attach" to the index term.

If the reference consists of more than one word, this opens the door for an off-by-one page number problem. This situation applies particularly to personal names. The following example will have a number that some may consider too low by one:

```
\index{Gauss, Carl Friedrich}%
```
Carl Friedrich⟨*page break*⟩Gauss

The `%` at the end of the line with the `\index` entry avoids a space, so the entry is tightly attached to "Carl". But, by just plain bad luck, the page break occurred before "Gauss". If there is no other occurrence of "Gauss" on the cited page, it may not be obvious to the reader that "Carl" is the target. Therefore, the "safe" approach is to attach the index term to the word that a reader is most likely to look for. For this example, the following input would be safer:

```
Carl Friedrich Gauss%
\index{Gauss, Carl Friedrich}
```
This will attach the index entry to "Gauss".

Index terms referring to phrases

Index entries for phrases, like names, will benefit from consideration of where a reader will look in the text for a match.

If an important phrase consists of several words, it may be desired to index it in multiple ways. In such a case, it is appropriate to attach the different index entries to different words in the phrase to allow for the possibility that a page break might occur somewhere in the middle of the phrase. The following example illustrates this situation:

```
\index{Dual Basis Lemma}%
Dual Basis Lemma%
\index{Lemma!Dual Basis}
```
Observe that there is no `%` following the second index entry. This prevents the word "Lemma" from running together with whatever text follows.

4. Sorting problems

Here are some easy rules to remember about the makeindex sort:

- only entries that match *exactly* will be grouped;
- differences in spacing will prevent a match;
- an entry with a space at the beginning will sort before anything else;
- accented letters won't match unaccented letters;
- entries that start with math coding will not sort with the alphabetical entries.

Spacing anomalies resulting from careless input

Unlike TEX, makeindex doesn't compress spaces or ignore initial spaces. So these four entries in the source file will yield different results in the sorted `.ind` file, and thus four separate — but apparently identical — entries in the printed index:

```
\index{multiple words}
\index{ multiple words}
\index{multiple  words}
\index{multiple words }
```
The first version (with the fewest spaces) is the best form to use.

Up to three levels are permitted per entry, and since makeindex matches the entire input string, the content of all levels must be identical. Like the examples above, the two- and three-level entries below will not be sorted together, as shown following the input:

```
\index{first level!second level}
\index{ first level !second  level}
\index{first level!second level!third level}
\index{ first level !second  level!third level}
\index{first level!second  level! third level}
```

Output:

first level
    second level, 99
        third level, 99
first level
    second level
        third level, 99
    second level, 99
        third level, 99

Spacing anomalies resulting from index entries in "moving arguments"

Another, less obvious, occasion for mismatched spaces can occur when an index term is entered in a caption, a section heading, a footnote, or other "moving argument" or an environment in which TEX commands become "fragile". If the index term consists only of text, there is no problem. However, if the index term contains a command (such as `\textit`), the environment processing automatically adds a space following the command. No error or warning is issued, and the anomaly will only be noticeable in the index output. Usually, it should be possible to move the index input outside the restricted environment, but if this isn't possible, the only workable repair is to precede the command within the index entry by the command `\string`, for example,

```
\index{De Revolutionibus@\string\textit{De Revolutionibus}}
```

This isn't documented in any of the "usual places", but was reported to `LaTeX-bugs` in 1995 (https://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex%2F1435`latex/1435`), and answered with the recommendation given here, as an attempt to repair it within LATEX is likely to cause worse problems than it fixes.

See also the subsection "Index terms referring to displayed material" on page 5.

Spacing anomalies in text resulting from index input

While we're on the subject of spaces, it's worth pointing out that, although TEX compresses multiple spaces in text, the spaces must be consecutive. Multiple index entries may be entered on separate lines, for example:

```
Lord Kelvin%
\index{Thomson, William|seeonly{Kelvin, Lord}}%
\index{Kelvin, Lord (William Thomson)}
```

If the `%`s are omitted from the ends of all three lines, those line endings will be interpreted as spaces that are not consecutive. TEX will not compress such spaces, and the result will be uneven spacing in the output, possibly an occasional space at the beginning or end of a line (leaving a ragged margin), or even an unwanted

page break with an off-by-one page number. So spaces in text as well as in index entries *do* matter.

Accents in index entries

Whether an accented letter is entered as a combination of control sequence and letter or as a single UTF-8 character, an accented "a" will not match an unaccented one when sorted by makeindex. To obtain the proper alphabetical (or other desired) order, a sort key must be provided in addition to the string to be printed. The form of such an entry is the following:

```
\index{⟨sort key⟩@⟨print string⟩}
\index{Erdos, Pal@Erd\H{o}s, P\'al}
```

with the output

Erdős, Pál

Such a sort key can be provided at any level of an entry.

5. Math in index entries

Whether "mathy" entries are listed separately or integrated with the main alphabetical index, sort keys should be used to establish logical relative positioning of these entries. For example, an entry in the alphabetical index for "L^p-spaces" entered as

```
\index{L p spaces@$L^p$-spaces}
```

would be sorted at the beginning of the "L"s. If entered as

```
\index{Lp spaces@$L^p$-spaces}
```

it would be sorted after entries beginning with "Lo".

The first important point is that all index terms that start with a math expression should be given a sort key. Even if you want to keep all math entries together, if sort keys are not provided, the result is a hodgepodge, greatly reducing the value of the index to the user.

The second, and most important point, is that you should decide on a system and stick to it; be consistent.

If you decide to separate a list of notation from the other index material, there are some alternatives worth considering. If the notation list will contain descriptions as well as the symbols themselves, two packages which provide facilities for such presentation are glossaries [Ta] and nomencl [VS]. Both packages have good documentation, and they will not be discussed further here. The following notes assume that makeindex is used directly.

Where to sort math entries

If many of your index entries are math expressions rather than text, it is worth considering whether they should be folded in, listed as a separate group at the beginning, or even segregated in a separate "Notation Index".

If a separate index is desired, the mechanics of multiple indexes are covered in Section 7.

If math entries are included in a single index, the suggestions above should suffice to help decide how to set up the ones that begin with letters for logical sorting. Math entries that begin with symbols (and ones for which no sort key has been provided) will always be sorted in a separate group at the beginning of the index;

the order will be that of an ASCII sort of the unmodified input strings, which is not particularly intuitive.

How to sort math entries

The first principle here is to make it easy for a reader to find a particular term; an unordered (or randomly ordered) list is usually not the best approach.

Unless you have a clear concept of how the math entries should be organized, don't include a sort key for the first run. Instead, produce the initial `.idx` file and examine what's there to help decide on an appropriate arrangement.

If a large proportion of the math entries start with a letter (perhaps in several different styles), an alphabetic arrangment, either with mixed styles or with each style set separately, might be considered. That is, A, B, C, $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, $\mathcal{A}$, $\mathcal{B}$, ..., vs. A, $\mathfrak{A}$, $\mathcal{A}$, B, $\mathfrak{B}$, $\mathcal{B}$, C, $\mathfrak{C}$, ...

Entries beginning with numerals, e.g., 3-manifold, are sometimes sorted with symbols, and sometimes "alphabetically" with the numeral placed as if the word were spelled out, "three-manifold". The choice is largely subjective, and depends on the nature of the subject matter.

Similarly, the placement of entries beginning with Greek letters depends on the subject matter. If there are only a few, including an entry "δ-function" with the letter D, either at the beginning of that letter group, or placed as if "delta" were spelled out, may be appropriate. In either case, an appropriate sort field is needed:

```
\index{d function@$\delta$-function}
\index{delta function@$\delta$-function}
```

Entries that begin with non-alphabetic or numeric symbols are best grouped so that all instances of the same first symbol are together and entries within that group arranged in an order that a reader will find logical. These smaller groups can then be arranged appropriately.

The next step is to assign sort keys that will guide makeindex in sorting the entries into the desired order. For convenience in keeping track of the keys, you might want to print out a copy of the index as generated from entries before sort keys are assigned and write the keys onto that list as you assign them. When inputting the keys to the file, be sure to insert them identically for all affected entries.

For keys, some authors have simply used ascending numbers, making sure that all have the same number of digits and perhaps leaving gaps to allow for additions. Or each subgroup can be assigned a number and the remainder of the sort key provided by a "math-free" form of the entry content.

For a separate symbols index, alphabetic keys can be assigned, followed by some sub-ordering mechanism; as in the alphabetic index, a gap will be inserted in the output between different initial sort letters.

Once all math entries have been assigned suitable keys, go back into the source file(s) and add the keys to the `\index` entries; then proceed with the usual processing cycle.

6. **Locating problem entries**

Problem entries can be of several kinds:

- entries rejected by the makeindex program, reported in the `.ilg` file;

- multiple seemingly identical entries in the index output (see "Spacing anomalies" above);
- entries in the output that appear to be out of order;
- entries that fail when LaTeXing the `.ind` file.

Errors and warnings reported in the `.ilg` file

`Extra '`x`' at position ... in line ...`

This is probably the most common error, where x is one of these characters used for special purposes in the `.idx` file:

- `@` (to separate a sort key from the printable entry),
- `!` (to separate an index term into different levels),
- `|` (to substitute for `\` in certain commands that are to be passed along to the `.ind` file indicating special treatment by makeindex, e.g., `|(` and `|)` that indicate the beginning and end of a page span).

Care must be taken to avoid using these characters in the text of an index entry.

An `@` usually gets into an index entry unexpectedly, when a fragile command expands to an internal form (where `@` is treated as a letter) as it is written into the `.idx` file. The solution is to insert the command `\protect` just before the fragile command, for example:

`\index{index term@\protect\`⟨*fragilecommand*⟩`}`

This is rare, and usually occurs in a math expression, either when an author has been particularly inventive in defining new notation or as the result of using a front-end preprocessor. Needless to say, commands like this should not be used in the part of the index entry that is to be sorted.

The `|` might appear in a math expression, $|A|$ (`$|A|$`). The equivalent command `\vert` should be used instead in the index entry to replace `|`: `$\vert A\vert$`. Similarly, `\|` or `||` (as in $\|A\|$, `$\|A\|$` or `$||A||$`) should be replaced by its equivalent `\Vert`.

An `!` might represent "factorial" in a math expression; there is no predefined command to replace this, but a suitable definition could be added in the preamble for use in this situation:

```
\newcommand\factorial{\mathclose!}
```

An exclamation mark is also used for a negative thin space (`\!`); the substitute command, if needed, is `\negthinspace`.

Warnings about extra range opening/closing operators

These have the form

`## Warning (input =` x`.idx, line = ...; output =` x`.ind, line = ...):`
`-- Extra range opening operator (.`

or

`## Warning (input =` x`.idx, line = ...; output =` x`.ind, line = ...):`
`-- Extra range closing operator ).`

where x is the file name. While entries reported in these warnings are usually "harmless", there is a bug in makeindex that can result in a problem when the `.ind` file is read by LaTeX.

If two entries with exactly the same entry text and the command to open a page range (`|(`) are not separated by a matching entry specifying the close of the page range (`|)`), makeindex will convert the `|(` of the second instance to `\(` in the `.ind`

file, followed by the page number in braces. LaTeX will interpret the `\(` as the beginning of an in-line math expression, and will generate an error

```
! Missing $ inserted.
```

It is not easy to recover from or bypass this error in the LaTeX run where the error is reported. The best way to avoid the problem is to resolve all warnings about "extra range opening operators" before proceeding. Delete the `.ind` file before rerunning LaTeX until these warnings are gone, and only then permit the `.ind` file to be included in the LaTeX run.

(The maintainer of makeindex has been notified, and is looking into the problem. These notes will be amended when the problem is fixed.)

Problems *not* reported in the `.ilg` file

`\index` terms are entered verbatim into the `.idx` file. If a misspelled command is typed into an index entry, it will not be detected until the `.ind` file is read back in. The error will be the obvious

```
! Undefined control sequence.
```

and the line number shown refers to the `.ind` file.

A more insidious problem occurs if a comment is inserted, starting with `%`, within an index entry. (It would have to be entered on more than one line to avoid an error while processing the `.tex` file.) In this case, the `%` is copied along with the rest of the text of the entry into one line in the `.idx` file, where it is sorted without reformatting by makeindex. When the resulting `.ind` file is read by LaTeX the `%` is interpreted as the beginning of a comment, and the remainder of the entry is "lost". This might not even result in an error message, but if one appears, it is sure to be confusing. If there is no error message, the result will be a truncated or an out-of-order entry in the index, which could be hard to find. A quick check in the `.idx` file for instances of `%` is not a bad idea.

Problems not noticed until the `.ind` file is processed by LaTeX

Usually, errors are detected by makeindex, but occasionally, something gets through to cause a problem while LaTeX is processing the `.ind` file.

The error

```
! Missing $ inserted.
```

can occur when two entries for the same opening page range are not separated by a matching entry for a closing range, as in

```
entry\index{entry|(}
...
entry\index{entry|(}
...
entry\index{entry|)}
```

A warning is given in the `.ilg` file in this situation, as mentioned above (page 10).

7. Multiple indexes

Support for multiple indexes is provided by the imakeidx package [BG]. This package has several particularly attractive features:

- The conventional LaTeX option mechanism is used to identify the intended index. If no option is specified, then the main index is assumed. This, in

turn, makes it possible to delay the decision whether to have one or more than one index.
- In many cases, it is possible to have the index(es) produced in the same run as the main document, without running makeindex separately. (See below for details and exceptions.)
- With the exception of the optional argument on the `\index` entries, there is no difference in how index entries are entered.

This recommendation to use the imakeidx package supersedes the former recommendation for the package amsmidx. Other packages for preparing multiple indexes are not compatible with AMS document classes.

If it seems that multiple indexes would be useful, the division should be scoped out before any index terms are added to the document text. However, the decision to actually produce more than one index can be delayed. Say a separate author index may be wanted. Instead of identifying all index entries simply by `\index`, use (for example) `\authindex` for those that would be indexed separately, and define the new command (in the preamble) as

```
\let\authindex\index
%\newcommand{\authindex}{\index[authors]}
```

Then enter the candidates for an author index as, for example,

```
... Lord Kelvin,\authindex{Kelvin, Lord} ...
```

For the initial processing, all index entries will be combined into a single index. Then, if the decision is made to actually create a separate index, comment out the `\let` definition and remove the `%` from the `\newcommand` line. The entries in the file will already be identified appropriately.

imakeidx options

After this package has been loaded with `\usepackage{makeidx}`, some options need to be specified to set up the individual indexes. These instructions are typical.

```
\makeindex[name=auth, title={Author index}
\makeindex[name=subj, title={Subject index}
```

The value assigned to `name` specifies the name of the `.idx` file, here `auth.idx` and `subj.idx`. (These will be converted later by makeindex into corresponding `.ind` files.) The `title` option provides the text for the title and running heads of that index.

One more option may be needed: `original`. This option will suppress the one-pass processing of the index. In this case, it will be necessary to perform the makeindex conversion explicitly, either manually or by means of a script; instructions for doing this can be found elsewhere.

Problems and workarounds

There are a couple of problems that (currently) prevent the full power of imakeidx from being used:

- With AMS document classes, the running head on the final page of an index is just "Index" instead of the specified title. This is not a problem if the `[original]` option is used, but it means that the index files must be processed separately, and the last page of the index isn't balanced. There *is* a workaround for this problem; see below.

- With some AMS book series packages that have unusual chapter first page styling, the output format is corrupted. Again, the `[original]` option avoids the problem and restores the correct first-page formatting at the expense of convenience and a balanced last page. As of this writing, there is no workaround.

  Affected book series, at present, are those with the document classes `gsm-l`, `stml-l`, and `amstext-l`.

Fixing the running heads for one-pass processing

The same command `\indexname` is used by both imakeidx and the AMS document classes to specify the running head text. The `title` option to `\makeindex` sets up the desired title, but (as of the 2013 version of imakeidx) this is reset to the default value "Index" before the last page of the index is completed. By reiterating the setting of `\indexname` just before the affected index is invoked, this problem is overcome. An example:

```
\renewcommand{\indexname}{Author Index}
\printindex[auth]
```

8. Processing shortcuts

While making adjustments to individual index entries, or to experiment with different arrangements of, say, a symbols index, you may want to avoid reprocessing the entire source document. In such a case it may be convenient to work directly in the `.idx` file. (Of course, you must remember to make all final changes in the main source files.)

If chapters are called in from a master file with `\include` commands, the following procedure can shorten the testing cycle.

- Copy the master file to another name, say `test-index.tex`, and insert the line `\includeonly{}` to suppress input of the text files.
- If you have a large bibliography, it is probably worth commenting out the commands that cause it to be processed.
- Copy the `.idx` file using the same name as the new "test" master file.
- Modify entries in the test `.idx` file as desired.
- `makeindex test-index`
- `latex test-index`

Adaptation for multiple indexes should be reasonably straightforward.

The output will contain only the index(es) and other elements triggered by commands other than `\include`, such as `\tableofcontents` and bibliography commands (if you haven't commented them out).

Reiterate this process until you are happy with the result. Then compare the modified `.idx` file with the original, and post the changes to the affected `\index` entries in the text files.

9. An alternate approach for a list of notation

When one of the "indexes" is a list of notation, it may be more appropriate to think of it as a kind of glossary, rather than an index. The approach described here assumes that you are *not* using one of the packages mentioned earlier (page 8).

This approach is perhaps best implemented using a tabular style across the full width of the page, rather than the usual two-column presentation of the index

format. For maximum flexibility, it is suggested that the list be prepared by hand, which allows direct control over the ordering of the entries, as opposed to adhering to the rather strict constraints of an ASCII sort. It also permits inclusion of additional useful material, such as a (brief) description of the terms listed.

Page numbers for such a list are provided using `\label` at the point in the text where the term first appears (or at its most important use or definition) and `\pageref{`⟨*label*⟩`}` in the separate list.

Here is one possible format.

******* to be completed *******

10. Creating and processing an index manually

For some books, it is not practical to generate an index by entering `\index{...}` terms directly in source files, and the decision is made to compile the index by hand. (One reason might be that only `.pdf` files are available for some pages.)

The recommended method for doing this is to enter index terms, in page number order, into a file using the same format as the `.idx` file. This can then be sorted into alphabetical order using makeindex, the same method used to order index data created in the "usual" way.

Keep the `.idx` file in page number order. It will be easier to find entries that need corrections if typos are found or if changes in the text result in changed page numbers.

The `.idx` file

Entries in an `.idx` file have this format:

`\indexentry{`*term to be indexed*`}{`*page number*`}`

Multi-level terms and terms that require special sort directives are entered using the same notation as if they were entered directly in the document source file; see page 2 for examples.

The makeindex program can handle page numbers that are integral digits and ordinary lowercase roman numerals (the paging usually used for material in the frontmatter), but special instructions (encoded in `.ist` files; not described here) are required for anything more complicated. If any special page number handling is required, request assistance from AMS mailto:tech-support@ams.org`tech-support`.

Processing the `.idx` file into an `.ind` file

The `.ind` file is what will be read in by `\printindex`. An `.idx` file is processed into the `.ind` file by this command:

`makeindex` ⟨*filename1*⟩`.idx` ⟨*filename2*⟩`.idx ...`

Multiple `.idx` files can be included in the same sort by listing their names on the same input line, separated by spaces. The output file will have the name ⟨*filename1*⟩`.ind`; the log file will be ⟨*filename1*⟩`.ilg`. Check the log for errors, and make corrections in the `.idx` files as necessary.

The `.ind` file

The format of the `.ind` file produced by makeindex is the following:

```
\begin{theindex}

\item aaa, 99
```

```
\item aba, 23, 76

\indexspace

\item baa, 42
\item bcd
  \subitem cde, 57
  \subitem cdf, 92
...

\end{theindex}
```

The driver file

A file to generate the printed index using the `.ind` file that results from this manual procedure can be very simple, assuming that nothing else is included and the final output format is a book. Name this file ⟨*filename1*⟩`.tex`:

```
\documentclass{amsbook}
\begin{document}
\setcounter{page}{⟨desired starting page number⟩}
\printindex
\end{document}
```

Using `amsbook` as the class, the starting page number should be odd, since the default is `[openright]`; specifying an even page number will result in a blank (numbered) page preceding the index.

To be continued ***

Possible enhancements (all require testing):

- How to add "headings" in the index, similar to the "random" headings in the index here. Include tactics for dealing with hyperref and omitting page numbers.
- Tactics for indexing a multi-volume work, where "page numbers" need to include volume information as well as actual number.
- UTF-8 input with makeindex.
- Use of xindy.

References

[BG] Claudio Beccari and Enrico Gregorio, *The package imakeidx*, v1.3b, 2016/04/0.

[Gr] George Grätzer, *More math into LaTeX*, fourth ed., Springer, New York, 2007.

[La] Leslie Lamport, *LaTeX: A document preparation system*, second revised ed., Addison-Wesley, Reading, MA, 1994.

[MG] Frank Mittelbach, Michel Goossens, et al., *The LaTeX companion*, second ed., Addison-Wesley, Reading, MA, 2004.

[Ta] Nicola L.C. Talbot, *User manual for glossaries.sty v4.21*, 2016/01/24.

[VS] Boris Veytsman, Bernd Schandl, Lee Netherton and CV Radhakrishnan, *nomencl – A package to create a nomenclature*, v4.2, 2005/09/22.

Index