

The AMS Export BibTeX style [`amsxport`]

American Mathematical Society
Michael Downes
David M. Jones

Version 2.00, 2004/06/28

Contents

1	Introduction	1
2	Implementation	2
3	Macros for month names	3
4	Formatting author names and editor names	10
5	Handling year, month, date	10
6	Journal abbreviations	11
7	Type info	12
8	Miscellaneous tail-end information	12
9	Wrapper for each entry	13
10	Entry types	13
11	Sorting the entries	16
12	Final pass for output	21
13	Journal abbreviations	22
14	To Do	26

1 Introduction

`amsxport` is the collective name for a collection of BibTeX styles that form a bridge between BibTeX and `amsrefs`. It produces output in which the structural information of the original BibTeX database file is retained. No ad hoc visual formatting is introduced that would hamper the application of alternative design specifications when rendering the information on paper or screen. This makes it possible to have BibTeX handle only extraction and sorting and leave the visual appearance to be controlled entirely by L^AT_EX. The process of applying typical bibliography specs to exported data is illustrated in the `amsrefs` package. The authoritative description of the export format is in the `amsrefs` documentation, inasmuch as interpreting the format is the chief task of the `amsrefs` package.

Each of the five `.bst` files generated from this file corresponds to a combination of `amsrefs` package options, as follows:

amsrn: This is the default style, used when none of the following is appropriate.

amsru: This is used when the `citation-order` option is specified. It is identical to `amsrn` except that the bibliography items are not sorted.

amsra: This is when with the `alphabetic` option is specified, byt the `y2k` option is *not* also specified.

amsry: This is used when both the `alphabetic` and `y2k` options are specified.

amsrs: This is used when the `shortalphabetic` option is specified.

Readers may wish to refer also to the standard BibTeX documentation (`btxdoc.tex`, `btxhak.tex`, `btxbst.doc`) for background information.

2 Implementation

The list below gives the fields that are expected to occur. (Other fields will be discarded without comment.) suggested fields.

```

1 <*bst>
2 ENTRY
3 {
4     address      institution  number      status
5     archive      isbn        organization  subtitle
6     author       issn        pages         title
7     booktitle    journal     part         translator
8     chapter      key         pii          type
9     date         language    preprint     url
10    edition      meeting     publisher    volume
11    editor       month       review       xid
12    eprint       mrnumber    school       year
13    howpublished note        series
14 }
15 { }
16 { label }
```

Some comments about specific fields:

archive: The archive that holds the eprint listed in the **eprint** field.

author: BibTeX doesn't allow this field (or any field) to be used more than once, but in the output its contents will be split into a separate key-value pair for each name. Likewise for **editor** and **translator**.

crossref: This is a built-in field type, so it doesn't occur in the argument of ENTRY above.

date: This is a generalization of the **year** and **month** fields. Its value should be written in ISO 8601 format, e.g., 1987-06-05; but the day and month are omissible, so this can be easily be used instead of the **year** field.

edition: For books. The BibTeX documentation suggests that the value of this field should be an ordinal word such as "Second". This may be because the BibTeX language provides only the weakest sort of support for an *is-numeric* test.

eprint: Electronic preprint information such as for www.arXiv.org.

institution: What's the difference between an **institution** and an **organization**? The BibTeX documentation says that **institution** should be used for technical reports and **organization** for other entry types.

isbn: International Standard Book Number.

issn: International Standard Serial Number. See the discussion of journal abbreviations.

language: Language of the work. This is especially desirable when the value of the title field is a translation of the true title (most often when the original language is one like Arabic or Chinese that poses rendering problems for non-native software).

meeting: Since BibTeX doesn't allow a field name to be the same as a function name, we have to use **meeting** instead of **conference** if we want to give the title of a conference in a dedicated field.

mrnumber: An alias for **review** which we provide because MathSciNet has been using it in the BibTeX output that it provides for a couple of years already.

organization: See the comment for **institution**.

pii: Publisher Item Identifier

part: This is for a long journal article that is published in separate issues of the same journal. There should be a separate BibTeX entry for each part (though all the ones after part 1 could crossref the first one).

preprint: It is not possible quite yet to assume that all preprints should be classified as eprints. If you have a preprint that is not an eprint, use this field to give the “address” where the preprint may be found (institution, preprint number, etc.).

review: Review numbers or similar pointers, e.g., for Mathematical Reviews or Zentralblatt.

school: The analog of “organization”, for a thesis.

status: Typically used for notes such as “to appear” or “in preparation” with journal articles.

subtitle: Typically used with a multipart journal article to give a subtitle for each part.

translator: This needs no explanation, surely? Except perhaps to note that the standard BibTeX styles don’t provide it.

url: Universal Resource Locator.

xid: This is used by a cross-referenced item to pass its identity to child entries that refer to it. It would not be necessary if BibTeX left the **crossref** field value accessible, but for some reason that I have not found yet (skimming in the source code) it is cleared internally before there is any opportunity to save it.

Practically speaking: For every **crossref** in your **.bib** files, the target should contain a matching **xid** field.

More on extra fields. In the Shelah bibliography, the **author** field is given without any accents in the author names, and the normal TeX version of the author names is given in a separate field called **trueauthor**.

Also, for MR reviews:

```
review = {MR 48:3735},
```

And for each author a country is indicated with the **fromwhere** field, e.g.,

```
fromwhere = {UK, IL},
```

This means the first author is from the United Kingdom and the second one from Israel.

3 Macros for month names

```
17 MACRO {jan} {"-01"}
18 MACRO {feb} {"-02"}
19 MACRO {mar} {"-03"}
20 MACRO {apr} {"-04"}
21 MACRO {may} {"-05"}
22 MACRO {jun} {"-06"}
23 MACRO {jul} {"-07"}
24 MACRO {aug} {"-08"}
25 MACRO {sep} {"-09"}
26 MACRO {oct} {"-10"}
27 MACRO {nov} {"-11"}
28 MACRO {dec} {"-12"}
```

Some scratch variables and global variables.

```
29 STRINGS { p s t l }
```

```

30 INTEGERS { len ptr }
31 <bst>

```

Note: BibTeX 0.99 requires a blank line in certain contexts, such as after the READ command, and will choke if it is missing. When it is necessary to add a blank line at the boundary of a macro code section we use a <bst> marker.

Some utility functions.

```

32 FUNCTION {NOT} { { #0 } { #1 } if$ }
33
34 FUNCTION {AND} { 'skip$ { pop$ #0 } if$ }
35
36 FUNCTION {OR} { { pop$ #1 } 'skip$ if$ }
37
38 FUNCTION {TRUE} { #1 }
39
40 FUNCTION {FALSE} { #0 }
41 <bst>

```

incr.ptr

```

42 FUNCTION {incr.ptr} { ptr #1 + 'ptr := }
43 <bst>

```

current.char Returns: first character of STRING

```

44 FUNCTION {current.char} { ptr #1 substring$ }

```

copy.two.chars The function copy.two.chars appends the next two characters from s to STRING.

Arg: STRING (partial copy of pages string)

Returns: Modified version of STRING

Side effects: advances ptr

```

45 FUNCTION {copy.two.chars} { t * incr.ptr s current.char * incr.ptr }
46 <bst>

```

is.digit

```

47 FUNCTION {is.digit} {
48   chr.to.int$
49   duplicate$
50   "0" chr.to.int$ #1 - >
51   swap$
52   "9" chr.to.int$ #1 + <
53   AND
54 }
55 <bst>

```

is.lowercase.letter In a previous implementation the is.letter function first lowercased the character using change.case\$, then tested it against the range a–z. But change.case\$ issues a warning if it is applied to a lone brace character.

```

56 FUNCTION {is.lowercase.letter} {
57   chr.to.int$ duplicate$
58   "a" chr.to.int$ #1 - >
59   swap$
60   "z" chr.to.int$ #1 + <
61   AND
62 }

```

is.uppercase.letter

```
63 FUNCTION {is.uppercase.letter} {
64     chr.to.int$ duplicate$
65     "A" chr.to.int$ #1 - >
66     swap$
67     "Z" chr.to.int$ #1 + <
68     AND
69 }
```

is.letter

```
70 FUNCTION {is.letter} {
71     duplicate$
72     is.lowercase.letter
73     { pop$ TRUE }
74     { is.uppercase.letter }
75     if$
76 }
```

skip.accent This implementation of skip.accent simply skips ahead one character whenever a backslash is found. This handles things like \v, \u, \k, \r. Although this is overly simplistic it is still an improvement over the previous method which was to not even make the attempt. [mjd,2001-10-27]

```
77 FUNCTION {skip.accent} { pop$ incr.ptr }
```

strip.label

```
78 FUNCTION {strip.label} {
79     's :=
80     #1 'ptr :=
81     ""
82     { s ptr #1 substring$ duplicate$ empty$ NOT }
83     { duplicate$ is.letter
84         { * }
85         { duplicate$ "\" =
86             { skip.accent }
87             'pop$
88             if$
89         }
90         if$
91         incr.ptr
92     }
93     while$
```

When the loop ends we have an extra empty-string on top of the stack to get rid of.

```
94     pop$
95 }
```

sortify

```
96 FUNCTION {sortify} {
97     purify$
98     "l" change.case$
99 }
```

rpad Warning: The use of text.length\$ here and in lpad restricts their use to strings that we know do not contain any special characters. Since we only use lpad for field names and rpad for purify\$ed strings, this is ok.

```

    <string> <int> rpad
100 FUNCTION {rpad} {
101     swap$
102     'l :=
103     l text.length$ - 'len :=
104     { len #0 > }
105     {
106         l " " * 'l :=
107         len #1 - 'len :=
108     }
109     while$
110     l
111 }

```

```

lpad <string> <int> lpad
112 FUNCTION {lpad} {
113     swap$
114     'l :=
115     l text.length$ - 'len :=
116     { len #0 > }
117     {
118         " " l * 'l :=
119         len #1 - 'len :=
120     }
121     while$
122     l
123 }

```

field.or.null

```

124 FUNCTION {field.or.null} {
125     duplicate$ empty$
126     { pop$ "" }
127     'skip$
128     if$
129 }

```

missing.or.empty

```

130 FUNCTION {missing.or.empty} { duplicate$ missing$ swap$ empty$ OR }
131 <bst>

```

ndash.and.skip.hyphens The function `append.ndash.and.skip.hyphens` adds `\ndash` to `STRING` when a hyphen is found and advances `ptr` to the next non-hyphen character in `s`.

Arg: `STRING` (partial copy of pages string)

Returns: modified version of `STRING`

Side effects: advances `ptr`

```

132 FUNCTION {append.ndash.and.skip.hyphens} {
133     "\ndash " *
134     incr.ptr
135     { s current.char "-" = }
136     { incr.ptr }
137     while$
138 }

```

n.dashify In the `n.dashify` function we store the given string in `s`, push an empty string on the stack, and start examining the characters of `s`. For a non-hyphen character, we just append a copy of it to the top string. For a hyphen, we append `\ndash` to the top string and advance the pointer until we reach a non-hyphen character. And there is one more exceptional case: for a backslash, we copy two characters instead of one. This keeps us from erroneously translating `\-` to `\\ndash`. One would scarcely expect to see an instance of `\-` in a page number field but let's face it, in actual use nearly anything can and does happen.

Arg: STRING (value of `pages` field)

Returns: STRING (dashified version of input string)

```

139 FUNCTION {n.dashify} {
140     's :=
141     #1 'ptr :=
142     ""
143     { s current.char 't := t "" = NOT }
144     { t "\" =
145         { copy.two.chars }
146         { t "-" =
147             { append.ndash.and.skip.hyphens }
148             { t * incr.ptr }
149         if$
150     }
151     if$
152 }
153 while$
154 }
```

missing.warning Standard warning message for a missing or empty field. For the user we call any such field 'missing' without respect to the distinction made by BibTeX between missing and empty.

```

155 FUNCTION {missing.warning} {
156     "missing" swap$ * " in " * cite$ * warning$
157 }
158 <bst>
```

string.length Because BibTeX doesn't provide a straightforward `string.length` function (`text.length$` counts "special characters" as a single character), it appears necessary to implement one the hard way.

```

159 INTEGERS { string.ptr }
160
161 FUNCTION {string.length} {
162     #1 'string.ptr :=
163     { duplicate$ string.ptr #1 substring$ "" = 'FALSE 'TRUE if$ }
164     { string.ptr #1 + 'string.ptr := }
165     while$
166     pop$
167     string.ptr #1 -
168 }
```

format.title

```

169 FUNCTION {format.title}{
170     duplicate$
171     missing.or.empty
172     { pop$ "" }
```

```

173      { "t" change.case$ }
174      if$
175 }

```

start.field

```

176 FUNCTION {start.field} {
177     #12 lpad "=" * write$
178 }

```

fin.field

```

179 FUNCTION {fin.field} { "}," write$ newline$ }
180 <bst>

```

write.field

```

181 FUNCTION {write.field} {
182     duplicate$
183     missing$
184     { pop$ missing.warning }
185     { duplicate$ empty$
186       { pop$ missing.warning }
187       { swap$ start.field write$ fin.field }
188     } if$
189   }
190   if$
191 }

```

optional.field

```

192 FUNCTION {optional.field} {
193     duplicate$
194     missing.or.empty
195     { pop$ pop$ }
196     { swap$ start.field write$ fin.field }
197     if$
198 }

```

optional.title.field

```

199 FUNCTION {optional.title.field} {
200     duplicate$
201     missing.or.empty
202     { pop$ pop$ }
203     { format.title swap$ start.field write$ fin.field }
204     if$
205 }

```

optional.pages.field

```

206 FUNCTION {optional.pages.field} {
207     duplicate$
208     missing.or.empty
209     { pop$ pop$ }
210     { swap$ start.field
211       n.dashify
212       write$ fin.field
213     }
214     if$
215 }

```


optional.mr.field

```

216 FUNCTION {optional.mr.field} {
217     duplicate$
218     missing.or.empty
219     { pop$ pop$ }
220     {
221         swap$ start.field
222         "\MR{" swap$ * "}" * write$ fin.field
223     }
224     if$
225 }

```

remove.ordinal.suffix

```

226 FUNCTION {remove.ordinal.suffix} {
227     's :=
228     s string.length 'len :=
229     #1 'ptr :=
230     { ptr len < s ptr #1 substring$ is.digit AND }
231     'incr.ptr
232     while$
233     s ptr global.max$ substring$ 'p :=
234     p "st" =
235     { TRUE }
236     { p "nd" =
237         { TRUE }
238         { p "rd" =
239             { TRUE }
240             { p "th" =
241                 { TRUE }
242                 { FALSE }
243             }
244         }
245         if$
246     }
247     if$
248 }
249 if$
250 { s #1 ptr #1 - substring$ }
251 { s }
252 if$
253 }

```

optional.edition.field

```

254 FUNCTION {optional.edition.field} {
255     duplicate$ missing.or.empty
256     { pop$ pop$ }
257     { swap$ start.field
258         remove.ordinal.suffix
259         write$ fin.field
260     }
261     if$
262 }

```

4 Formatting author names and editor names

Take a name list in Bib_TE_X form (names separated by the word “and”) and output the desired form of each name.

```
263 INTEGERS { nameptr numnames namesleft }
264 STRINGS { namelist fieldname }
```

format.name The `format.name` function operates on a single name, producing a string of the form

von Last, First, Jr.

Args: *namelist*, *index* (top; integer)

Returns: formatted version of the *n*th name in *namelist* where integer *index* specifies *n*.

```
265 FUNCTION {format.name} { "{vv~}{ll}{, ff}{, jj}" format.name$ }
266 <bst>
```

optional.name.field The function `optional.name.field` handles the task of splitting a multiple-name field value into multiple fields with single values.

Args: *fieldname*, *namelist* (top)

Side effects: writes a key/value pair to the output file for each name in *namelist*.

```
267 FUNCTION {optional.name.field}{
268     duplicate$ missing.or.empty
269     { pop$ pop$ }
270     { swap$ 'fieldname :=
271       duplicate$ num.names$
272       'namesleft :=
273       #1 'ptr :=
274       { namesleft #0 > }
275       { fieldname start.field
```

Copy the name string before running `format.name` on it.

```
276         duplicate$ ptr format.name write$
277         fin.field
278         ptr #1 + 'ptr :=
279         namesleft #1 - 'namesleft :=
280     }
281     while$
```

At this point we have an extra copy of the name string on the stack.

```
282         pop$
283     }
284     if$
285 }
```

5 Handling year, month, date

assemble.date.field In the output we produce a `date` field instead of the year and month fields normally used in Bib_TE_X files. In the date field we use ISO date notation (e.g., 1987-06-05) to facilitate switching between full and abbreviated month names. The month and day parts are frequently absent, making the date field equivalent to a year field.

```
286 FUNCTION {assemble.date.field}
287 { date missing$
288   { year missing.or.empty
```

```

289         { status missing.or.empty
290           { "No year or other date information for "
291             cite$ * warning$ }
292           'skip$
293           if$
294           "status"
295         }
296         { year month missing$ { "" } { month } if$ * }
297       if$
298     }
299     { date }
300   if$

```

If the date string is now equal to “status”, it means that we have a status field or we already gave a warning about lack of date info; in either case omit the date field.

```

301   duplicate$ "status" =
302     { pop$ pop$ }
303     { swap$ start.field write$ fin.field }
304   if$
305 }

```

6 Journal abbreviations

Use of journal abbreviations is recommended to make supplying ISSN numbers easier. To use this feature, you need a STRING definition of the form

```

@STRING{cpam="cpam/0010-3640/"
        #"Communications in Pure and Applied Mathematics"}

```

Then when you write

```
journal=cpam,
```

exporting with `amsxport` will produce

```
journal={cpam},
ISSN={0010-3640},
```

The text before the ISSN number in the STRING definition is preferably a repetition of the abbreviation name (leaving expansion to be done on the `LATEX` side); or it could be the full journal name.

Warning: If you include braces and write `journal={cpam}`, `BibTEX` will not expand the abbreviation but leave it as the field value; and then you will not get automatic lookup of the ISSN number.

`optional.journal.field`

```

306 FUNCTION {optional.journal.field} {
307   duplicate$ missing.or.empty
308   { pop$ pop$ }
309   {
310     's :=
311     start.field
312     #1 'ptr :=
313     s current.char is.lowercase.letter
314     {
315       s "/" * 's :=
316       { s ptr #1 substring$ "/" = NOT }
317       { incr.ptr }
318     while$

```

Journal name started with a lowercase letter, but we couldn't find no slash? All right, just let it go through as it is.

```
319             s ptr #1 + #1 substring$ empty$
320             { s #1 ptr #1 - substring$ }
```

But if we did find a slash, then we probably have found us an ISSN number that we can write out.

```
321             {
322                 s #1 ptr #1 - substring$
323                 write$
324                 fin.field
```

ISSNs are always nine characters in length (four digits, hyphen, four digits).

```
325                 "ISSN" start.field
326                 s ptr #1 + #9 substring$
327             }
328             if$
329         }
330         { s }
331         if$
332         write$
333         fin.field
334     }
335     if$
336 }
```

7 Type info

`optional.type.field` Putz around with the type info a little, for theses mainly. Two types here: the type field, and the entry type.

```
337 FUNCTION {optional.type.field}
338 { duplicate$ missing.or.empty
339   { type$ "mastersthesis" = { "Master's Thesis" }
340     { type$ "phdthesis" = { "Ph.D. Thesis" }
341       { "" } if$ } if$
342     swap$ pop$
343   }
344   'skip$
345   if$
346   duplicate$ empty$
347   { pop$ pop$ }
348   { swap$ start.field write$ fin.field }
349   if$
350 }
```

8 Miscellaneous tail-end information

`url.note.status.review` The following items are common to all entry types are output at the tail end of the entry.

```
351 FUNCTION {url.note.status.review} {
352     "url" url optional.field
353     "note" note optional.field
354     "status" status optional.field
355     "review" review optional.field
```

The `review` field should perhaps be translated into multiple fields like author names, if more than one review is given.

Data from MathSciNet will have the Math Reviews number in an `mrnumber` field. We assume that a given entry will have a `review` field or an `mrnumber` field but not both.

```
356     "review" mrnumber optional.mr.field
357 }
```

9 Wrapper for each entry

`start.entry`

```
358 FUNCTION {start.entry} {
359     newline$
360     "\bib{" cite$ * "}{ " *
Merge Master's thesis and Ph.D. thesis into a single type.
361     type$ "mastersthesis" = type$ "phdthesis" = OR
362     { "thesis" }
363     { type$ }
364     if$
365     * "}{ " * write$
366     newline$
367 <*debug>
368     "sort.label" label sortify optional.field
369     "sort.key" sort.key$ optional.field
370 </debug>
371 }
```

`fin.entry`

```
372 FUNCTION {fin.entry} {
373     "}" write$
374     newline$
375 }
```

10 Entry types

Here are the types of entries that are normally allowed in a BibTeX file:

book: A book with an explicit publisher.

booklet: A work that is printed and bound, but without a named publisher or sponsoring institution.

inbook: A part of a book, which may be a chapter (or section or whatever) and/or a range of pages.

incollection: A part of a book having its own title.

manual: Technical documentation.

mastersthesis: A Master's thesis.

phdthesis: A PhD thesis.

proceedings: The proceedings of a conference.

techreport: A report published by a school or other institution, usually numbered within a series.

article: An article from a journal or magazine.

inproceedings: An article in a conference proceedings.

conference: An alias for **inproceedings**.

unpublished: A document having an author and title, but not formally published.

misc: Use this type when nothing else fits.

article A journal article differs from an inproceedings article by not having booktitle, publisher, editor, and other such info.

```

376 FUNCTION {article} {
377     start.entry
378     "author" author optional.name.field
379     "translator" translator optional.name.field
380     "title" title format.title write.field
381     "subtitle" subtitle optional.title.field
382     "language" language optional.field
383     "organization" organization optional.field
384     "how" howpublished optional.field
Construct a date value from date, year, month fields
385     "date" assemble.date.field
386     "ISSN" issn optional.field
387     "journal" journal optional.journal.field
388     "volume" volume optional.field
389     "number" number optional.field
390     "pages" pages optional.pages.field
391     "PII" pii optional.field
392     "archive" archive optional.field
393     "eprint" eprint optional.field
394     "preprint" preprint optional.field
395     url.note.status.review
396     fin.entry
397 }
```

inproceedings An inproceedings entry may have xid, booktitle, meeting, publisher info; it is not expected to have journal, eprint, or preprint info.

```

398 FUNCTION {inproceedings} {
399     start.entry
400     "author" author optional.name.field
401     "translator" translator optional.name.field
402     "title" title format.title write.field
403     "subtitle" subtitle optional.title.field
404     "language" language optional.field
405     "organization" organization optional.field
406     "how" howpublished optional.field
```

Construct a date value from date, year, month fields

```

407     "date" assemble.date.field
408     "xid" xid optional.field
```

The following fields might be inherited from the parent of an article in a proceedings volume or collection.

```

409     "conference" meeting optional.field
410     "booktitle" booktitle optional.title.field
```

Include the edition in case this is an inbook entry.

```

411     "edition" edition optional.edition.field
412     "editor" editor optional.name.field
413     "series" series optional.field
414     "volume" volume optional.field
415     "publisher" publisher optional.field
416     "address" address optional.field
417     "pages" pages optional.pages.field
```

```

418     url.note.status.review
419     fin.entry
420 }

```

inbook

```

421 FUNCTION {inbook} { inproceedings }

```

incollection

```

422 FUNCTION {incollection} { inproceedings }

```

inproceedings

```

423 FUNCTION {conference} { inproceedings }

```

book The book type includes a meeting field because otherwise we'd have to repeat the entire definition for the proceedings type with that as the sole difference.

```

424 FUNCTION {book} {
425     start.entry
426     "author" author optional.name.field
427     "editor" editor optional.name.field
428     "translator" translator optional.name.field
429     "title" title missing$ { booktitle } { title } if$
430     format.title write.field
431     "subtitle" subtitle optional.title.field
432     "type" type optional.type.field
433     "language" language optional.field
434     "conference" meeting optional.field
435     "edition" edition optional.edition.field
436     "series" series optional.field
437     "publisher" publisher optional.field
438     organization missing.or.empty
439     { "institution" institution optional.field }
440     { "organization" organization optional.field }
441     if$
442     "address" address optional.field
443     "how" howpublished optional.field
444     "date" year write.field
445     "volume" volume optional.field

```

The number here is intended for a tech report.

```

446     "number" number optional.field
447     "ISBN" isbn optional.field
448     url.note.status.review
449     fin.entry
450 }

```

booklet

```

451 {booklet}
452 FUNCTION {booklet} { book }

```

manual

```

453 FUNCTION {manual} { book }

```

mastersthesis

```

454 FUNCTION {mastersthesis} { book }
455

```

```

456 FUNCTION {phdthesis} { book }
457
458 FUNCTION {proceedings} { book }
459
460 FUNCTION {collection} { book }
461
462 FUNCTION {techreport} { book }
463
464 FUNCTION {unpublished} { book }
465
466 FUNCTION {misc} { book }
467
468 FUNCTION {default.type} { misc }
469 <bst>

```

BibTEX 0.99 keels over if READ does not have a blank line after it.

```

470 <bst>
471 READ
472 <bst>

```

11 Sorting the entries

Since the final labels will be produced by the `amsrefs` package, we don't need to worry about generating suffixes to disambiguate between equal stems. All we have to do is strive to generate stems that are identical to the ones that `amsrefs` will generate, so that `amsxport` will sort its output in a fashion that will be consistent with the labels.

Overall, this simplifies our job, since we don't have to keep track of duplicate labels, so we can process each entry in isolation. However, we do have to be more careful about the handling of text accents and text symbols in order to maintain compatibility with `amsrefs`.

See (*inter alia*) the sections “Lexical structure of names” and “Generating alphabetic labels” in the `amsrefs` implementation documentation for more information.

```

473 <*sort>

```

chop.word

```

474 FUNCTION {chop.word} {
475   's :=
476   'len :=
477   s #1 len substring$ =
478   { s len #1 + global.max$ substring$ }
479   's
480   if$
481 }

```

sort.format.title

```

482 FUNCTION {sort.format.title} {
483   't :=
484   "A " #2
485   "An " #3
486   "The " #4 t chop.word
487   chop.word
488   chop.word

```



```

489     sortify
490     #1 global.max$ substring$
491 }

```

format.lab.names

```

492 <*alpha | short>
493 FUNCTION {format.lab.names} {
494     's :=
495     s num.names$ 'numnames :=
496     numnames #1 >
497     { numnames #4 >
498         { #3 'namesleft := }
499         { numnames 'namesleft := }
500         if$
501         #1 'nameptr :=
502         ""
503         { namesleft #0 > }
504         { nameptr numnames =
505             { s nameptr "{ff }{vv }{ll}{ jj}" format.name$ "others" =
506                 { "{\etalchar{+}}" * }
507                 { s nameptr "{v{} }{l{} }" format.name$ * }
508                 if$
509                 }
510             { s nameptr "{v{} }{l{} }" format.name$ * }
511             if$
512             nameptr #1 + 'nameptr :=
513             namesleft #1 - 'namesleft :=
514             }
515             while$
516             numnames #4 >
517             { "{\etalchar{+}}" * }
518             'skip$
519             if$
520             }
521             { s #1 "{v{} }{l{} }" format.name$
522                 duplicate$ text.length$ #2 <
523                 <alpha> { pop$ s #1 "{ll}" format.name$ #3 text.prefix$ }
524                 <short> { pop$ s #1 "{ll}" format.name$ #1 text.prefix$ }
525                 'skip$
526                 if$
527             }
528             if$
529         }
530
531 FUNCTION {author.key.label}
532 { author empty$
533     { key empty$
534     { cite$ #1 #3 substring$ }
535     { key #3 text.prefix$ }
536     if$
537     }
538     { author format.lab.names }
539     if$
540 }

```

```

541
542 FUNCTION {author.editor.key.label}
543 { author empty$
544   { editor empty$
545   { key empty$
546     { cite$ #1 #3 substring$ }
547     { key #3 text.prefix$ }
548     if$
549   }
550 { editor format.lab.names }
551   if$
552   }
553   { author format.lab.names }
554   if$
555 }
556
557 FUNCTION {author.key.organization.label}
558 { author empty$
559   { key empty$
560 { organization empty$
561   { cite$ #1 #3 substring$ }
562   { "The " #4 organization chop.word #3 text.prefix$ }
563   if$
564 }
565 { key #3 text.prefix$ }
566   if$
567   }
568   { author format.lab.names }
569   if$
570 }
571
572 FUNCTION {editor.key.organization.label}
573 { editor empty$
574   { key empty$
575 { organization empty$
576   { cite$ #1 #3 substring$ }
577   { "The " #4 organization chop.word #3 text.prefix$ }
578   if$
579 }
580 { key #3 text.prefix$ }
581   if$
582   }
583   { editor format.lab.names }
584   if$
585 }
586
587 FUNCTION {calc.label}{
588   type$ "book" = type$ "inbook" = OR
589   'author.editor.key.label
590   { type$ "proceedings" =
591 'editor.key.organization.label
592 { type$ "manual" =
593   'author.key.organization.label
594   'author.key.label

```

```

595  if$
596 }
597     if$
598     }
599     if$
600     strip.label
601     sortify
602 <*alpha>
603     year field.or.null
604 <!y2k>     #3 #8 substring$
605     purify$ *
606 </alpha>
607     'label :=
608 }
609 </alpha | short>

```

sort.format.names

```

610 FUNCTION {sort.format.names} {
611     's :=
612     #1 'ptr :=
613     ""
614     s num.names$ 'numnames :=
615     numnames 'namesleft :=
616     { namesleft #0 > }
617     { ptr #1 >
618       { " " * }
619       'skip$
620       if$
621       s ptr "{ll}{ ff}{ jj}" format.name$ 't :=
622       ptr numnames = t "others" = AND
623       { "et al" }
624       { t sortify }
625       if$
626       #20 rpad *
627       ptr #1 + 'ptr :=
628       namesleft #1 - 'namesleft :=
629     }
630     while$
631 }

```

sort.warning

```

632 FUNCTION {sort.warning} {
633     "to sort, need " swap$ * " or key in " * cite$ * warning$
634 }

```

author.sort

```

635 FUNCTION {author.sort}
636 { author empty$
637   { key empty$
638     { "author" sort.warning
639       ""
640     }
641     { key sortify }
642   if$

```

```

643     }
644     { author sort.format.names }
645     if$
646 }

```

author.editor.sort

```

647 FUNCTION {author.editor.sort}
648 { author empty$
649   { editor empty$
650     { key empty$
651       { "author, editor," sort.warning
652         ""
653       }
654       { key sortify }
655     } if$
656   }
657   { editor sort.format.names }
658 } if$
659 }
660 { author sort.format.names }
661 if$
662 }

```

author.organization.sort

```

663 FUNCTION {author.organization.sort}
664 { author empty$
665   { organization empty$
666     { key empty$
667       { "author, organization," sort.warning
668         ""
669       }
670       { key sortify }
671     } if$
672   }
673   { "The " #4 organization chop.word sortify }
674 } if$
675 }
676 { author sort.format.names }
677 if$
678 }

```

editor.organization.sort

```

679 FUNCTION {editor.organization.sort}
680 { editor empty$
681   { organization empty$
682     { key empty$
683       { "editor, organization," sort.warning
684         ""
685       }
686       { key sortify }
687     } if$
688   }
689   { "The " #4 organization chop.word sortify }
690 } if$

```

```

691     }
692     { editor sort.format.names }
693     if$
694 }

```

presort

```

695 FUNCTION {presort} {
696   <*alpha | short>
697   calc.label
698   label #20 rpad
699   </alpha | short>
700   type$ "book" = type$ "inbook" = OR type$ "collection" = OR
701       'author.editor.sort
702   { type$ "proceedings" =
703       'editor.organization.sort
704       { type$ "manual" =
705           'author.organization.sort
706           'author.sort
707       }
708   }
709   if$
710 }
711 if$
712 <alpha | short>      * " " *
713 <!alpha | short>    year field.or.null sortify #12 rpad * " " *
714   title field.or.null sort.format.title *
715   #1 entry.max$ substring$
716   'sort.key$ :=
717 }

718 ITERATE {presort}
719
720 SORT
721 </sort>

```

12 Final pass for output

`write.with.newlines` The `write.with.newlines` functions stores a string in `s` and runs through it looking for occurrences of `^^M`; when one is found, the substring before it is written out with a newline, the remainder is left in `s`, and the cycle begins again.

```

722 FUNCTION {write.with.newlines}
723 { 's := #1 'ptr :=
724   { s ptr #2 + #1 substring$ "" = NOT }
725   { s ptr #3 substring$ "^^M" =
726       { s #1 ptr #1 - substring$ write$ newline$
727         ptr #3 + 'ptr :=
728         s ptr global.max$ substring$ 's :=
729         #1 'ptr :=
730       }
731       { incr.ptr }
732   }
733   if$
734   while$

```

```

735 s write$ newline$
736 }

begin.bin

737 FUNCTION {begin.bib} {
738     preamble$ empty$
739     'skip$
740     { preamble$ write.with.newlines }
741     if$
742     "% \bib, bibdiv, biblist are defined by the amsrefs package."
743     write$ newline$
744     "\begin{bibdiv}" write$ newline$
745     "\begin{biblist}" write$ newline$
746 }

end.bin

747 FUNCTION {end.bib} {
748     newline$
749     "\end{biblist}" write$ newline$
750     "\end{bibdiv}" write$ newline$
751 }

752 EXECUTE {begin.bib}
753
754 ITERATE {call.type$}
755
756 EXECUTE {end.bib}
757 </bst>

```

13 Journal abbreviations

By putting the ISSN into the L^AT_EX document, we make it easier to construct www queries from the bibliography data. These string definitions tie together journal abbreviations, ISSN numbers, and journal names.

These are the journals that are cited most frequently by American Mathematical Society authors at the present time (2000 CE).

The format used here is designed to make it easy for BibTeX to retrieve an ISSN number when given a journal abbreviation.

```

758 <*abbrevs>
759 @string{aa="aa/0065-1036/"
760 # "Acta Arithmetica"}
761 @string{actamath="actamath/0001-5962/"
762 # "Acta Mathematica"}
763 @string{asms="asms/0001-6969/"
764 # "Acta Scientiarum Mathematicarum"}
765 @string{advmath="advmath/0001-8708/"
766 # "Advances in Mathematics"}
767 @string{ajm="ajm/0002-9327/"
768 # "American Journal of Mathematics"}
769 @string{amm="amm/0002-9890/"
770 # "American Mathematical Monthly"}
771 @string{aifg="aifg/0373-0956/"
772 # "Annales de l'Institut Fourier"}
773 @string{am2="am2/0003-486X/Annals of Mathematics"}
774 @string{ap="ap/0091-1798/"

```

```

775  #"The Annals of Probability"}
776 @string{asens4="asens4/0012-9593/"
777  #"Annales Scientifiques de l'\{E}cole Normale Sup'\{e}rieure"}
778 @string{amb="amb/0003-889X/Archiv der Mathematik"}
779 @string{arma="arma/0003-9527/"
780  #"Archive for Rational Mechanics and Analysis"}
781 @string{bamsn="bamsn/0273-0979/"
782  #"Bulletin (New Series) of the American Mathematical Society"}
783 @string{bams="bams/0004-9727/"
784  #"Bulletin of the Australian Mathematical Society"}
785 @string{blms="blms/0024-6093/"
786  #"Bulletin of the London Mathematical Society"}
787 @string{bsmf="bsmf/0037-9484/"
788  #"Bulletin de la Soci'\{e}t'\{e} Math'\{e}matique de France"}
789 @string{craspi="craspi/0764-4442/"
790  #"Comptes Rendus de l'Acad'\{e}mie des Sciences (Paris)"
791  #" S'\{e}rie I Math'\{e}matique"}
792 @string{cjm="cjm/0008-414X/Canadian Journal of Mathematics"}
793 @string{cmb="cmb/0008-4395/"
794  #"Canadian Mathematical Bulletin"}
795 @string{colloqmath="colloqmath/0010-1354/"
796  #"Colloquium Mathematicum"}
797 @string{commalgebra="commalgebra/0092-7872/"
798  #"Communications in Algebra"}
799 @string{cmp="cmp/0010-3616/"
800  #"Communications in Mathematical Physics"}
801 @string{cpam="cpam/0010-3640/"
802  #"Communications on Pure and Applied Mathematics"}
803 @string{cmh="cmh/0010-2571/"
804  #"Commentarii Mathematici Helvetici"}
805 @string{cvta="cvta/0278-1077/"
806  #"Complex Variables"}
807 @string{compositiomath="compositiomath/0010-437X/"
808  #"Compositio Mathematica"}
809 @string{c="c/0010-485X/Computing"}
810 @string{constrapprox="constrapprox/0176-4276/"
811  #"Constructive Approximation"}
812 @string{dmj="dmj/0012-7094/"
813  #"Duke Mathematical Journal"}
814 @string{eraams="eraams/1079-6762/"
815  #"Electronic Research Announcements"
816  #" of the American Mathematical Society"}
817 @string{etds="etds/0143-3857/"
818  #"Ergodic Theory and Dynamical Systems"}
819 @string{fm="fm/0016-2736/"
820  #"Fundamenta Mathematicae"}
821 @string{gd="gd/0046-5755/"
822  #"Geometriae Dedicata"}
823 @string{illinoisjmath="illinoisjmath/0019-2082/"
824  #"Illinois Journal of Mathematics"}
825 @string{iumj="iumj/0022-2518/"
826  #"Indiana University Mathematics"}
827 @string{im="im/0020-9910/"
828  #"Inventiones Mathematicae"}

```

```

829 @string{israeljmath="israeljmath/0021-2172/"
830   #"Israel Journal of Mathematics"}
831 @string{iansm="iansm/??/"
832   #"Izvestiya Akademii Nauk SSSR. Seriya Matematicheskaya"}
833 @string{ja="ja/0021-8693/"
834   #"Journal of Algebra"}
835 @string{jams="jams/0894-0347/"
836   #"Journal of the American Mathematical Society"}
837 @string{jam="jam/0021-7670/"
838   #"Journal d'Analyse Math\`{e}matique"}
839 @string{jamm="jamm/0021-8928/"
840   #"Journal of Applied Mathematics and Mechanics"}
841 @string{jat="jat/0021-9045/"
842   #"Journal of Approximation Theory"}
843 @string{jamsa="jamsa/0263-6115/"
844   #"Journal of the Australian Mathematical Society"}
845 @string{jcta="jcta/0097-3165/"
846   #"Journal of Combinatorial Theory"}
847 @string{jde="jde/0022-0396/"
848   #"Journal of Differential Equations"}
849 @string{jdg="jdg/0022-040X/Journal of Differential Geometry"}
850 @string{jfa="jfa/0022-1236/"
851   #"Journal of Functional Analysis"}
852 @string{jllms2="jllms2/0024-6107/"
853   #"Journal of the London Mathematical Society"}
854 @string{jmaa="jmaa/0022-247X/"
855   #"Journal of Mathematical Analysis and Applications"}
856 @string{jmp="jmp/0022-2488/"
857   #"Journal of Mathematical Physics"}
858 @string{jmsj="jmsj/0025-5645/"
859   #"Journal of the Mathematical Society of Japan"}
860 @string{jnt="jnt/0022-314X/Journal of Number Theory"}
861 @string{jot="jot/0379-4024/"
862   #"Journal of Operator Theory"}
863 @string{jpaa="jpaa/0022-4049/"
864   #"Journal of Pure and Applied Algebra"}
865 % Braces instead of quotes, to hide the embedded \" control sequence
866 @string{jram={jram/0075-4102/}
867   #{Journal f\"{u}r die Reine und Angewandte Mathematik}}
868 @string{jssl="jssl/0022-4812/"
869   #"The Journal of Symbolic Logic"}
870 @string{laa="laa/0024-3795/"
871   #"Linear Algebra and its Applications"}
872 @string{mm="mm/0025-2611/"
873   #"Manuscripta Mathematica"}
874 @string{matzametki="matzametki/0025-567X/Matematicheskie Zametki"}
875 @string{ma="ma/0025-5831/"
876   #"Mathematische Annalen"}
877 @string{mc="mc/0025-5718/"
878   #"Mathematics of Computation"}
879 @string{mj="mj/0025-5513/"
880   #"Mathematica Japonica"}
881 @string{mn="mn/0025-584X/Mathematische Nachrichten"}
882 @string{mpcps="mpcps/0305-0041/"

```



```

883  #"Mathematical Proceedings of the"
884  #" Cambridge Philosophical Society"}
885 @string{mrl="mrl/1073-2780/"
886  #"Mathematical Research Letters"}
887 @string{ms="ms/0025-5521/"
888  #"Mathematica Scandinavica"}
889 @string{mathz="mathz/0025-5874/"
890  #"Mathematische Zeitschrift"}
891 @string{m="m/0025-5793/"
892  #"Mathematika"}
893 @string{mmj="mmj/0026-2285/"
894  #"The Michigan Mathematical Journal"}
895 @string{nmj="nmj/0027-7630/"
896  #"Nagoya Mathematical Journal"}
897 @string{na="na/0362-546X/Nonlinear Analysis"}
898 @string{nm="nm/0029-599X/Numerische Mathematik"}
899 @string{ojm="ojm/0030-6126/"
900  #"Osaka Journal of Mathematics"}
901 @string{pjm="pjm/0030-8730/"
902  #"Pacific Journal of Mathematics"}
903 @string{pams="pams/0002-9939/"
904  #"Proceedings of the American Mathematical Society"}
905 @string{pems2="pems2/0013-0915/"
906  #"Proceedings of the Edinburgh Mathematical Society"}
907 @string{pjaa="pjaa/0386-2194/"
908  #"Proceedings of the Japan Academy"}
909 @string{plms3="plms3/0024-6115/"
910  #"Proceedings of the London Mathematical Society"}
911 @string{qjmo2="qjmo2/0033-5606/"
912  #"The Quarterly Journal of Mathematics"}
913 @string{rmi="rmi/0213-2230/"
914  #"Revista Matem\`{a}tica Iberoamericana"}
915 @string{rmjm="rmjm/0035-7596/"
916  #"Rocky Mountain Journal of Mathematics"}
917 @string{sjam="sjam/0036-1399/"
918  #"SIAM Journal on Applied Mathematics"}
919 @string{sjc="sjc/0097-5397/"
920  #"SIAM Journal on Computing"}
921 @string{sjma="sjma/0036-1410/"
922  #"SIAM Journal on Mathematical Analysis"}
923 @string{sjna="sjna/0036-1429/"
924  #"SIAM Journal on Numerical Analysis"}
925 @string{sr="sr/0036-1445/"
926  #"SIAM Review"}
927 @string{sm="sm/0039-3223/"
928  #"Studia Mathematica"}
929 @string{tjm="tjm/0387-3870/"
930  #"Tokyo Journal of Mathematics"}
931 @string{ta="ta/0166-8641/"
932  #"Topology and its Applications"}
933 @string{t="t/0040-9383/"
934  #"Topology"}
935 @string{tams="tams/0002-9947/"
936  #"Transactions of the American Mathematical Society"}

```

937 `</abbrevs>`

14 To Do

—When there is no author or editor info to create a label from: Use the initial letters of all the words in the title. Plus the year. Maybe year first.

—Add `linelength` and `indent` options to control the output?

—Discuss `apacite` "originalfoo" fields etc.

—Discuss Beebe's work.

—Add bibliography.

—Add AUDIO, VIDEO entry types.

—Add a field for MR subject classification numbers and the like?

—Write a converter to convert from typical .bib data to preferred form.

Have to go via perl or something to capture STRING defs and PREAMBLE and interspersed comments? Or emacs lisp? Provide .bst version as fallback.

—Alpha labels: add numbers after sorting? subscript option? L3Y2a option.

—Test multi-author articles with ten-plus authors. What happens with the alpha labels then?

—For url field especially, check length and prebreak by hand instead of relying on BibTeX's slapdash stick-in-a-percent-at-column-72 regardless of where it happens to fall.

—How about a general 'contributor' field.

—Test with a wimpy older version of BibTeX.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	B	<code>\etalchar</code> ... 506, 517
<code>\"</code> ... 85, 144, 865, 867	<code>begin.bin</code> 737	
<code>\'</code> 777, 788,	<code>\bib</code> 360, 742	F
790, 791, 838, 914	<code>book</code> 424	<code>field.or.null</code> 124
<code>.bib</code> 3	<code>booklet</code> 451	<code>fin.entry</code> 372
<code>.bst</code> 1	<code>btxbst.doc</code> 1	<code>fin.field</code> 179
	<code>btxdoc.tex</code> 1	<code>format.lab.names</code> . 492
A	<code>btxhak.tex</code> 1	<code>format.name</code> 10, 10
<code>alphabetic option</code> . 1, 1	C	<code>format.name</code> 265
<code>amsrefs package</code> ...	<code>chop.word</code> 474	<code>format.title</code> 169
..... 1, 1, 1,	<code>citation-order op-</code>	<code>fromwhere</code> 3
1, 1, 16, 16, 16, 16	<code>tion</code> 1	I
<code>amsrn BibTeX style</code> ... 1	<code>conference</code> 2	<code>inbook</code> 421
<code>amsxport</code> 1, 11	<code>copy.two.chars</code> 4	<code>incollection</code> 422
<code>amsxport BibTeX style</code>	<code>copy.two.chars</code> 45	<code>incr.ptr</code> 42
..... 1, 16	<code>crossref</code> 3, 3	<code>inproceedings</code> 13
<code>append.ndash.and.skip.hyphens</code>	<code>current.char</code> 44	<code>inproceedings</code> 398, 423
..... 6	D	<code>institution</code> 2, 2, 2
<code>append.ndash.and.skip.hyphens</code> ■	<code>date</code> 10	<code>is.digit</code> 47
..... 132	E	<code>is.letter</code> 4
<code>article</code> 376	<code>editor</code> 2	<code>is.letter</code> 70
<code>assemble.date.field</code> 286	<code>editor.organization.sort</code>	<code>is.lowercase.letter</code> 56
<code>author</code> 3 679	<code>is.uppercase.letter</code> 63
<code>author.editor.sort</code> 647	<code>end.bin</code> 747	L
<code>author.organization.sort</code>	<code>eprint</code> 2	<code>lpad</code> 5, 5, 6
..... 663		<code>lpad</code> 112
<code>author.sort</code> 635		

- M**
- manual [453](#)
 - mastersthesis [454](#)
 - meeting [2](#)
 - missing.or.empty . [130](#)
 - missing.warning .. [155](#)
 - month [2](#)
 - \MR [222](#)
- N**
- n.dashify [7](#)
 - n.dashify [139](#)
 - \ndash [7](#), [133](#)
- O**
- optional.edition.field
..... [254](#)
 - optional.field ... [192](#)
 - optional.journal.field
..... [306](#)
 - optional.mr.field [216](#)
 - optional.name.field [10](#)
 - optional.name.field [267](#)
 - optional.pages.field
..... [206](#)
 - optional.title.field
..... [199](#)
 - optional.type.field [337](#)
 - organization [2](#), [2](#)
- P**
- pages [7](#)
 - presort [695](#)
- R**
- READ [4](#)
 - remove.ordinal.suffix
..... [226](#)
 - review [2](#), [12](#)
 - rpadd [5](#), [6](#)
 - rpadd [100](#)
- S**
- s [6](#), [7](#), [7](#), [21](#), [21](#)
 - shortalphabetic op-
tion [1](#)
 - skip.accent [5](#)
 - skip.accent [77](#)
 - sort.format.names [610](#)
 - sort.format.title [482](#)
 - sort.warning [632](#)
- T**
- sortify [96](#)
 - start.entry [358](#)
 - start.field [176](#)
 - string.length [7](#)
 - string.length [159](#)
 - strip.label [78](#)
 - translator [2](#)
 - trueauthor [3](#)
- U**
- url.note.status.review
..... [351](#)
- W**
- write.field [181](#)
 - write.with.newlines [21](#)
 - write.with.newlines [722](#)
- X**
- xid [3](#)
- Y**
- y2k option [1](#), [1](#)
 - year [2](#), [2](#)