The `bibexport.sh` script

Nicolas Markey

2011/11/28

Abstract

`bibexport.sh` is a small shell script, relying on BibTEX, that extracts entries of one or several `.bib` file(s). It will expand abbreviations and cross-references, except standard month and journal abbreviations. The output is indented as neatly as possible, yielding a readable `.bib` file even if the original file is not.

1 Exporting `.bib` files

1.1 Why and how?

BibTEX aims at allowing for the use of one single `.bib` file, containing many entries, from which BibTEX extracts only the `\cite`d ones. When sending a document to someone else, this requires either sending the whole file, or extracting the `\cite`d entries from the `.bib` file.

BibTEX also has a mechanism for using abbreviations and cross-references. When extracting entries of a large `.bib` file, it can be interesting to develop those abbreviations, in order to get a clean, self-contained `.bib` file. Also, it may be useful to develop cross-references in a `.bib` file, independently of any document.

`bibexport` can either extract entries that are cited in a document, or all the entries of one or several `.bib` files. It will always develop cross-references and abreviations, except standard abbreviations for months or some journals, that are defined in standard BibTEX styles. This script uses BibTEX. This has both pros and cons:

+ it is very simple. Basicaly, the script simply calls BibTEX, and the `.bst` file just outputs the name and the content of each field.

+ since it uses BibTEX, we are sure that it will handle everything "properly", *i.e.* in the same way as they will be handled when cited in a LATEX document;

= BibTEX has some strict limitations (especially "no more than 78 consecutive non-space characters") that we must be aware of. On the other hand, any such problem occuring within the script would also occur when compiling a document;

- abbreviations and cross-references will *always* be developped. It could be argued that this is also a positive point, but having the choice would be better.
- Many people seem to find BibTEX's internal language clumsy, and thus the script could be difficult to adapt to special needs. However, this is not *that* difficult, as will be explained later on. In the present case, adding more fields to be exported is quite easy.

1.2 Related scripts

Several other tools exist for achieving this task:

- `aux2bib`, written by Ralf Treinen, relies on `bib2bib`, which is a CAML program for selecting some entries in one or several `.bib` files. It does not expand anything, but includes all the necessary definitions and entries.
- `bibextract.sh`, by Nelson Beebe. This script uses AWK for extracting some entries out of a `.bib` file. It is said not to be compliant with cross-references.
- subset.bst, by David Kotz. export.bst develops the same ideas (but I discovered that only later on). subset.bst does not handle `@preamble`, neither does it "protect" standard abbreviations.

1.3 Some examples

- extracting `\cite`d references of a document, also including cross-references:

  ```
  bibexport.sh -o <result>.bib <file>.aux
  ```

- extracting `\cite`d references of a document, without crossrefs, and using a special `.bst` file:

  ```
  bibexport.sh -b <style>.bst -o <result>.bib <file>.aux
  ```

- export all the entries of two `.bib` files (including crossrefed entries):

  ```
  bibexport.sh -a -o <result>.bib <file1>.bib <file2>.bib
  ```

- export all the entries of two `.bib` files (without crossrefs):

  ```
  bibexport.sh -a -n -o <result>.bib <file1>.bib <file2>.bib
  ```

  In fact, the only difference between this and the previous one is that `crossref` field will be filtered out at the end of the script.

- export all the entries of two `.bib` files, using an extra file containing cross-referenced entries (which should not be included):

  ```
  bibexport.sh -a -e <crossref>.bib -n -o <result>.bib \
          <file1>.bib <file2>.bib
  ```

1.4 Exporting extra fields

By default, `bibexport.sh` exports only "standard" fields (those defined and used in plain.bst), as well as a few others. It is very easy to modify it in order to export other fields: it suffices to modify export.bst as follows:

- in the `ENTRY` list, add the name of the field you would like to export. Notice that `ENTRY` takes three space-separated lists as arguments; you must add extra fields in the first argument (actually, the last two are empty).
- in the function `entry.export.extra`, add a line of the form

  ```
  "myfield" myfield field.export
  ```

  where `myfield` is the name of the extra field you want to export.

Acknowledgements

I thank Éric Colin de Verdière, Richard Mathar, Harald Hanche-Olsen and Damien Pollet for suggesting several improvements or corrections.

2 The code

2.1 The shell script

2.1.1 Initialization

`testfiles` We check that the `.bst` files have the correct version number:

```
⟨*script⟩
function checkversion()
{
  kpsewhich expcites.bst > /dev/null ||
    echo -e "-----------\n--Warning-- file expcites.bst not found.\n-----------"
  grep -q $VDATE ‘kpsewhich expkeys.bst‘ ||
    echo -e "-----------\n--Warning-- the version of the .bst files does not match with that
}
⟨/script⟩
```

`usage` We first define how the script should be used:

```
⟨*script⟩
function usage()
{
echo "bibexport: a tool to extract BibTeX entries out of .bib files.
usage: ‘basename $0‘ [-h|v|n|c|a|d|s|t] [-b|e|es|ec|o|r file] file...

Basic options:
--------------
 -a, --all                  export the entire .bib files
 -o bib, --output-file bib  write output to file       [default: bibexport.bib]
```

```
 -t, --terse                operate silently
 -h, --help                 print this message and exit
 -v, --version              print version number and exit

Advanced options:
-----------------
 -b bst, --bst bst          specifies the .bst style file [default: export.bst]
 -c, --crossref             preserve crossref field               [default: no]
 -n, --no-crossref          remove crossref'd entries             [default: no]
 -e bib, --extra bib        extra .bib file to be used (crossrefs and strings)
 -es bib, --extras bib      extra .bib file to be used (for strings)
 -ec bib, --extrac bib      extra .bib file to be used (for crossrefs)
 -p, --preamble             write a preamble at beginning of output
 -r bib, --replace bib      replace .bib file(s) in the .aux file
 -d, --debug                create intermediate files but don't run BibTeX";
exit 0;
}
⟨/script⟩
```

opttoolate

We also have a function to warn if extra options are given after the names of input files, which is not allowed.

```
⟨*script⟩
function opttoolate()
{
if [ ${TOOLATE} -ne 0 ]; then
    echo "No option is allowed after the input files";
    exit 0;
fi
}
⟨/script⟩
```

VERSION
VDATE
ALL
CREF
DEBUG
FILE
EXT
EXTRA
EXTRABIB
REPLACEBIB
NEWBIB
SPACE
BST
TERSE
BANNER
ARGS
TOOLATE

We define the default value of some variables:

- `$VERSION`: the version number;
- `$VDATE`: the release date;
- `$ALL`: a flag indicating that all entries of the given (`.bib`) file are to be exported;
- `$CREF`: the value of `-min-crossrefs`;
- `$FILE`: the input file(s);
- `$EXT`: the extension (`.aux` or `.bib`) of input files;
- `$EXTRA`: list of possible extra `.bib` files without extension;
- `$EXTRABIB`: list of possible extra `.bib` files with extension;
- `$REPLACEBIB`: flag indicating that we will replace the `.bib` file given in the `.aux` file with a new one;

- `$NEWBIB`: new `.bib` file to replace that fiven in the `.aux` file;
- `$SPACE`: file name separator (can be ␣, comma or empty);
- `$BST`: the `.bst` file to be used;
- `$TERSE`: run silently;
- `$BANNER`: don't print the initial comment;
- `$ARGS`: the list of aruments passed to `bibexport.sh`;
- `$TOOLATE`: options are not allowed once we have encountered the first non-option argument.
- `$DEBUG`: create intermediate files but do not run BibTeX.

```
⟨*script⟩
## Version number
VERSION="3.01";
## Release date
VDATE="2011/11/28";

# ALL is a flag set to 1 when '-a' is given
ALL="0";
# FILE will be the main input file(s) (.aux or .bib, depending on '-a')
FILE="";
# EXT is the extension of the input file(s) (.aux, or .bib if '-a')
EXT=".aux";
# EXTRA and EXTRABIB are two copies of the extra files ('-e'), used to
# include crossref'd entries and @string's
EXTRA="";
EXTRABIB="";
# REPLACEBIB ('-r') is set to 1 when the \bibdata of the .aux input file
# must be ignores (then '-e' must be used)
REPLACEBIB="0";
# NEWBIB will contain the argument given to -r
NEWBIB="";
# BST is the .bst file to be used (default to export.bst)
BST="export";
# TERSE will be set to '-terse' if '-t' is given
TERSE="";
# BANNER is used to turn on or off the preamble informations in the output
BANNER="false";
# CREF is the number of citations of crossrefs from which the crossref'd entry
# must be included.
CREF="0";

# SPACE will be either ' ' or ','
SPACE="";
# TOOLATE is used to prevent extra options after the main file
```

```
TOOLATE="0";
# DEBUG is used to create files but not run BibTeX.
DEBUG="";

ARGS=$@;
⟨/script⟩
```

2.1.2 Handling arguments

If no argument have been supplied, we call `usage`. Otherwise, we check version number.

```
⟨*script⟩
if [ $# -eq 0 ]; then
  usage;
fi
checkversion;
⟨/script⟩
```

Otherwise, we enter a `while`-loop for handling the whole list of arguments:

```
⟨*script⟩
while [ $# != 0 ]; do
    case $1 in
⟨/script⟩
```

- `-a` or `--all`: export all the bibliography. This means that we input `.bib` files.

```
⟨*script⟩
        -a|--all)
    ## - export all entries in the input file(s)
    ## - the input files are BibTeX files
            opttoolate;
            EXT=""; SPACE=""; ALL=1;
            shift ;;
⟨/script⟩
```

- `-b` or `--bst`: specifies the style file. It seems that BibTEX does not like the `./`*`style`*`.bst` syntax, and we have to handle that case separately.

```
⟨*script⟩
        -b|--bst)
    ## - specifies the .bst file to use (default to 'export.bst')
            opttoolate;
            if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bst`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bst`";
            fi
            BST="${DOLLARTWO}";
            shift 2;;
⟨/script⟩
```

- `-d` or `--debug`: only creates (and preserves) the intermediate files. This can help finding problems with the script or `.bst` files.

```
⟨*script⟩
        -d|--debug)
    ## - debug mode: we create files but do not run bibtex
    ## - instead, we print what we would have done...
            opttoolate;
    DEBUG="echo";
    shift ;;
⟨/script⟩
```

- `-e` or `--extra`: when we want to export all the entries of a `.bib` file, we can specify an extra `.bib` file that would contain entries that we don't want to export, but that are needed, *e.g.* for crossrefs.

```
⟨*script⟩
        -e|--extra)
    ## - extra input files (containing crossrefs or strings)
    ## - they will be included twice: once before the main file(s)
    ##   (for @string's), once after (for crossrefs). We fool BibTeX
    ##   by naming the first one 'file.bib' and the second one
    ##   'file.bib.bib', to avoid complains.
            opttoolate;
            if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bib`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bib`";
            fi
            EXTRA="${EXTRA}${DOLLARTWO},";
            EXTRABIB="${EXTRABIB},${DOLLARTWO}.bib";
            shift 2;;
⟨/script⟩
```

- `-es` or `--extras`: if, for some reason, including extra files twice is not possible, this options provides a way of including extra `.bib` files only before the main `.bib` file(s).

```
⟨*script⟩
        -es|--extras)
    ## - extra input files (containing strings)
    ## - will be included *before* the main files (hence not suitable
    ##   for crossrefs)
            opttoolate;
            if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bib`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bib`";
            fi
            EXTRA="${EXTRA}${DOLLARTWO},";
            shift 2;;
```

```
⟨/script⟩
```

- `-ec` or `--extrac`: similar to te previous one, but for file(s) included after the main `.bib` file(s).

```
⟨*script⟩
        -ec|--extrac)
    ## - extra input files (containing crossrefs)
    ## - will be included only *after* the main files (hence not
    ##   suitable for @string's)
            opttoolate;
            if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bib`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bib`";
            fi
            EXTRABIB="${EXTRABIB},${DOLLARTWO}.bib";
            shift 2;;
⟨/script⟩
```

- `-o` or `--output`: the name of the output file.

```
⟨*script⟩
        -o|--output-file)
    ## - name of the output file
    ## - we force it to end with '.bib'
            opttoolate;
            if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bib`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bib`";
            fi
            OUTPUT="${DOLLARTWO}.bib";
            shift 2 ;;
⟨/script⟩
```

- `-c` or `--crossref` (or others): this options means that we want crossrefs to be included. Note that for any entry, field inheritage will be performed.

```
⟨*script⟩
        -c|--crossref|--crossrefs|--with-crossref|--with-crossrefs)
    ## - whether or not to preserve 'crossref' keys.
    ## - by default, they are removed, but crossref'd entries are
    ##   included.
    ## - crossrefs are *always* expanded anyway.
            opttoolate;
            CREF="1" ;
            shift ;;
⟨/script⟩
```

- `-n` or `--no-crossref`: don't include crossref'ed entries.

```
⟨*script⟩
        -n|--no-crossref|--without-crossref|--no-crossrefs|--without-crossrefs)
    ## - to remove crossref'd entries (hence remove 'crossref' keys).
            opttoolate;
            CREF="20000" ;
            shift ;;
⟨/script⟩
```

- `-r` or `--replace`: this provides a way of replacing the `.bib` files given by `\ibdata` in the `.aux` file with (a) new one(s).

```
⟨*script⟩
        -r|--replace)
    ## - to replace the file(s) given in \bibdata in the .aux file with
    ##   (a) new one(s).
            opttoolate;
    REPLACEBIB="1";
    if [ "`dirname $2`" = "." ]; then
                DOLLARTWO="`basename $2 .bib`";
            else
                DOLLARTWO="`dirname $2`/`basename $2 .bib`";
            fi
            NEWBIB="${NEWBIB}${DOLLARTWO}.bib,";
            shift 2;;
⟨/script⟩
```

- `-v` or `--version` for version number:

```
⟨*script⟩
        -v|--version)
            echo "This is bibexport v${VERSION} (released ${VDATE})"; exit 0;;
⟨/script⟩
```

- `-p` or `--preamble` for inserting some informations at the beginning of the output file:

```
⟨*script⟩
        -p|--preamble|--with-preamble)
            BANNER="true";
            shift ;;
⟨/script⟩
```

- `-t` or `--terse` for asking BibTEX to run silently:

```
⟨*script⟩
        -t|--terse|--silent)
            TERSE=" -terse ";
            shift ;;
⟨/script⟩
```

- other dash-options are erroneous (except `-h`, but...):

```
    ⟨*script⟩
            -*)
                usage;;
    ⟨/script⟩
```

- there should only remain file names: we add those names to the list of files.

```
    ⟨*script⟩
            *)
        ## - list of input files
        ## - we ensure that no extra option is given later...
        TOOLATE="1";
                if [ "`dirname $1`" = "." ]; then
                    DOLLARONE="`basename $1 ${EXT}`";
                else
                    DOLLARONE="`dirname $1`/`basename $1 ${EXT}`";
                fi
                FILE="${FILE}${SPACE}${DOLLARONE}${EXT}";
                if [ ${ALL} -eq 1 ]; then
                    SPACE=",";
        else
    SPACE=" ";
                fi;
                shift;;
    ⟨/script⟩
```

That's all folks:

```
⟨*script⟩
    esac
done
⟨/script⟩
```

2.1.3 The core of the script

We first set the name of the result and intermediary files:

```
⟨*script⟩
FINALFILE=${OUTPUT};
if [ ! "${FINALFILE}" ]; then
    FINALFILE="bibexport.bib";
fi
TMPFILE="bibexp.`date +%s`";
⟨/script⟩
```

We then create the `.aux` file for the main run of BibTEX. Note that this could call BibTEX, with the expkeys.bst file, in the case where we want to export all entries of a `.bib` file but not crossrefs. Note how, in that case, we trick BibTEXfor inputing extra files twice: we include then with their short name first (with no extension), and then with the full name. We *need* to do that, since `string` abbreviations must be defined first, while crossrefs must occur after having been referenced.

```
⟨*script⟩
if [ -z "${EXT}" ]; then ## we export all entries
    if [ -z "${EXTRA}" ]; then ## we have no extra files
        cat > ${TMPFILE}.aux <<EOF
\citation{*}
\bibdata{${FILE}}
\bibstyle{${BST}}
EOF
    else ## we have extra files (e.g. for crossrefs) but want all entries from ${FILE}
   ## we first extract the keys to be used:
        cat > ${TMPFILE}.aux <<EOF
\citation{*}
\bibdata{${FILE}}
\bibstyle{expkeys}
EOF
        ## This run may generate errors. We redirect the output:
        bibtex -min-crossrefs=${CREF} -terse ${TMPFILE} >/dev/null 2>&1;
        mv -f ${TMPFILE}.bbl ${TMPFILE}.aux;
## and then prepare the .aux file for exporting:
        cat >> ${TMPFILE}.aux <<EOF
\bibdata{${EXTRA}${FILE}${EXTRABIB}}
\bibstyle{${BST}}
EOF
    fi
else ## we only export entries listed in the given .aux file:
  if [ ! "x${REPLACEBIB}" = "x1" ]; then
    cat ${FILE} | sed -e "s/bibstyle{.*}/bibstyle{${BST}}/" > ${TMPFILE}.aux;
  else
    cat ${FILE} | sed -e "s/bibstyle{.*}/bibstyle{${BST}}/" \
      -e "s/bibdata{.*}/bibdata{${EXTRA}${NEWBIB%,}${EXTRABIB}}/" > ${TMPFILE}.aux;
  fi
fi
⟨/script⟩
```

This was the hard part. We now call BibTEX, clean and rename the output file, and remove intermediary files:

```
⟨*script⟩
if [ -z "$DEBUG" ]; then
    bibtex -min-crossrefs=${CREF} ${TERSE} ${TMPFILE};
    if [ -e ${FINALFILE} ]; then
mv ${FINALFILE} ${FINALFILE}-save-`date "+%Y.%m.%d:%H.%M.%S"`
    fi
    echo "" > ${FINALFILE}
else
    echo "bibtex -min-crossrefs=${CREF} ${TERSE} ${TMPFILE};"
fi
if [ ! "${BANNER}" = "false" ]; then
    ## list of cited entries
    if [ -z "$DEBUG" ]; then
sed -i -e "s/\\\bibstyle{.*}/\\\bibstyle{expcites}/" ${TMPFILE}.aux
```

```
mv ${TMPFILE}.aux ${TMPFILE}-cites.aux
bibtex -terse -min-crossrefs=${CREF} ${TMPFILE}-cites
echo -ne "@comment{generated using bibexport:\n" >> ${FINALFILE};
echo -ne "  creation date:\t`date +\"%c\"`\n" >> ${FINALFILE};
echo -ne "  command:\t\t`basename $0` ${ARGS}\n" >> ${FINALFILE};
if [ -z "${EXT}" ]; then
    echo -ne "  source files:\t\t${FILETAB}\t\t\t${EXTRABIBTAB}\n" >> ${FINALFILE}; \
fi
cat ${TMPFILE}-cites.bbl >> ${FINALFILE};
echo -ne "  bibexport-version:\tv${VERSION} (${VDATE})\n" >> ${FINALFILE};
echo -ne "  bibexport-maintainer:\tmarkey(at)lsv.ens-cachan.fr\n" >> ${FINALFILE};
sed -i -e "s/}/)/g" ${FINALFILE};
echo -n -e "}\n\n\n" >> ${FINALFILE};
rm -f ${TMPFILE}-cites.bbl ${TMPFILE}-cites.aux ${TMPFILE}-cites.blg
    fi
fi
if [ ${CREF} -ne 1 ]; then
    if [ -z "$DEBUG" ]; then
egrep -iv '^ *crossref *= *[^,]+,?$' \
    ${TMPFILE}.bbl >> ${FINALFILE};
    else
echo "egrep -iv '^ *crossref *= *[^,]+,?$' ${TMPFILE}.bbl >> ${FINALFILE};"
    fi
else
    if [ -z "$DEBUG" ]; then
cat ${TMPFILE}.bbl >> ${FINALFILE};
    else
echo "cat ${TMPFILE}.bbl >> ${FINALFILE};"
    fi
fi
if [ -z "$DEBUG" ]; then
    rm -f ${TMPFILE}.bbl ${TMPFILE}.aux ${TMPFILE}.blg;
else
    echo "rm -f ${TMPFILE}.bbl ${TMPFILE}.aux ${TMPFILE}.blg";
fi
⟨/script⟩
```

2.2 The **expkeys.bst** file

The only role of that file is to export the list of entries to be exported. It is used when we export all the entries of `.bib` files, except those of *extra* `.bib` files. Thus:

```
⟨*expkeys⟩
ENTRY{}{}{}
READ
FUNCTION{export.key}
{
  "\citation{" cite$ "}" * * write$ newline$
}
ITERATE{export.key}
```

```
⟨/expkeys⟩
```

2.3 The expcites.bst file

This file is used for exporting and formating the list of `\cite`d entries. We begin with some parameters defining the margins

2.3.1 Some configuration values

`left.width` `right.width` `url.right.width` `left.short.width` `right.short.width` `left.delim` `right.delim`

```
⟨*expcites⟩
FUNCTION{left.width}{#23}
FUNCTION{right.width}{#55}
FUNCTION{url.right.width}{#61}
FUNCTION{left.short.width}{#10} %% for @preamble
FUNCTION{right.long.width}{#63}
FUNCTION{left.delim}{quote$}
FUNCTION{right.delim}{quote$}
⟨/expcites⟩
```

2.3.2 Entries

We only want to export `\cite`d keys, so we won't use any field.

`ENTRY`

```
⟨*expcites⟩
ENTRY{dummy}{}{}
⟨/expcites⟩
```

2.3.3 Basic functions

`or` `and` `not`

```
⟨*expcites⟩
FUNCTION{not}
{
    {#0}
    {#1}
  if$
}
FUNCTION{and}
{
    'skip$
    {pop$ #0}
  if$
}
FUNCTION{or}
{
    {pop$ #1}
    'skip$
  if$
}
⟨/expcites⟩
```

2.3.4 Splitting strings

We design functions for splitting strings, so that the final `.bib` file will be cleanly indented.

`space.complete`
`split.string`

```
⟨*expcites⟩
INTEGERS{left.length right.length}
STRINGS{ s t }
INTEGERS{bool}
FUNCTION{space.complete}
{
  'left.length :=
  duplicate$ text.length$ left.length swap$ -
  {duplicate$ #0 >}
    {
      swap$ " " * swap$ #1 -
    }
  while$
  pop$
}
FUNCTION{split.string}
{
  'right.length :=
  duplicate$ right.length #1 + #1 substring$ "" =
    {""}
    {
      's :=
      right.length
      {duplicate$ duplicate$ s swap$ #1 substring$ " " = not and}
        {#1 -}
      while$
      duplicate$ #2 <
        {
          pop$ "    " s * ""
        }
        {
          duplicate$ s swap$ #1 swap$ substring$
          swap$
          s swap$ global.max$ substring$
        }
      if$
    }
  if$
}
⟨/expcites⟩
```

2.3.5 Exporting cited entries

Now we initialize, and export `\cited` entries.

init.cited.keys
write.cited.keys
write.cited.keys.last
cited.keys
end.cited.keys

```
⟨*expcites⟩
FUNCTION{init.cited.keys}
{
  left.delim 's :=
  #0 'bool :=
}
FUNCTION{write.cited.keys}
{
  bool
    {"" left.width space.complete swap$}
    {"  cited keys: " left.width space.complete swap$
     #1 'bool :=}
  if$
  {duplicate$ text.length$ right.width >}
    {
      right.width split.string 't :=
      *
      write$ newline$
      "" left.width space.complete t
    }
  while$
  pop$ pop$ t
}
FUNCTION{write.cited.keys.last}
{
  bool
    {"" left.width space.complete swap$}
    {"  cited keys: " left.width space.complete swap$
     #1 'bool :=}
  if$
  {duplicate$ duplicate$ text.length$ #1 substring$ "," = not}
    {duplicate$ text.length$ #1 - #1 swap$ substring$}
  while$
  duplicate$ text.length$ #1 - #1 swap$ substring$
  right.delim *
  {duplicate$ "" = not}
    {
      right.width split.string 't :=
      *
      write$ newline$
      "" left.width space.complete t
    }
  while$
  pop$ pop$
}
```

```
FUNCTION{cited.keys}
{
  s cite$ ", " * *  's :=
  s text.length$ #4000 >
    {s write.cited.keys 's :=}
    'skip$
  if$
}
FUNCTION{end.cited.keys}
{
  s write.cited.keys.last
}
⟨/expcites⟩
```

2.3.6 Now, we export...

We now export everything...

```
⟨*expcites⟩
FUNCTION{article}{cited.keys}
FUNCTION{book}{cited.keys}
FUNCTION{booklet}{cited.keys}
FUNCTION{conference}{cited.keys}
FUNCTION{habthesis}{cited.keys}
FUNCTION{inbook}{cited.keys}
FUNCTION{incollection}{cited.keys}
FUNCTION{inproceedings}{cited.keys}
FUNCTION{journals}{cited.keys}
FUNCTION{manual}{cited.keys}
FUNCTION{mastersthesis}{cited.keys}
FUNCTION{misc}{cited.keys}
FUNCTION{phdthesis}{cited.keys}
FUNCTION{proceedings}{cited.keys}
FUNCTION{techreport}{cited.keys}
FUNCTION{unpublished}{cited.keys}
READ
EXECUTE{init.cited.keys}
ITERATE{cited.keys}
EXECUTE{end.cited.keys}
⟨/expcites⟩
```

2.4 The **export.bst** file

2.4.1 Some configuration values

`left.width` `right.width` `url.right.width` `left.short.width` `right.short.width` `left.delim` `right.delim`

We define here the indentation values, and the field delimiters. *short* width are used for `@preamble`.

```
⟨*export⟩
FUNCTION{left.width}{#18}
FUNCTION{right.width}{#55}
```

```
FUNCTION{url.right.width}{#61}
FUNCTION{left.short.width}{#10} %% for @preamble
FUNCTION{right.long.width}{#63}
FUNCTION{left.delim}{"{"}
FUNCTION{right.delim}{"}"}
%FUNCTION{left.delim}{quote$}
%FUNCTION{right.delim}{quote$}
⟨/export⟩
```

2.4.2 Entries

We use standard entries here. Of course, more entries could be added for special `.bib` files. Those extra entries will also have to be added in the main exporting function.

`ENTRY`

```
⟨*export⟩
ENTRY{
% Standard fields:
    address
    author
    booktitle
    chapter
    edition
    editor
    howpublished
    institution
    journal
    key
    month
    note
    number
    organization
    pages
    publisher
    school
    series
    title
    type
    volume
    year
% Special (but still somewhat standard) fields (natbib, germbib, ...):
    abstract
    doi
    eid
    isbn
    issn
    language
    url
}{}{}
```

```
⟨/export⟩
```

2.4.3 Basic functions

No comment.

or
and
not

```
⟨*export⟩
FUNCTION{not}
{
    {#0}
    {#1}
  if$
}
FUNCTION{and}
{
    'skip$
    {pop$ #0}
  if$
}
FUNCTION{or}
{
    {pop$ #1}
    'skip$
  if$
}
⟨/export⟩
```

2.4.4 Splitting strings

We design functions for splitting strings, so that the final `.bib` file will be cleanly indented.

space.complete
split.string
split.url
split.name

```
⟨*export⟩
INTEGERS{left.length right.length}
STRINGS{ s t }
FUNCTION{space.complete}
{
  'left.length :=
  duplicate$ text.length$ left.length swap$ -
  {duplicate$ #0 >}
    {
      swap$ " " * swap$ #1 -
    }
  while$
  pop$
}
FUNCTION{split.string}
{
  'right.length :=
  duplicate$ right.length #1 + #1 substring$ "" =
    {""}
    {
      's :=
      right.length
      {duplicate$ duplicate$ s swap$ #1 substring$ " " = not and}
        {#1 -}
      while$
      duplicate$ #2 <
        {
          pop$ "    " s * ""
        }
        {
          duplicate$ s swap$ #1 swap$ substring$
          swap$
          s swap$ global.max$ substring$
        }
      if$
    }
  if$
}
FUNCTION{split.url}
{
  'right.length :=
  duplicate$ right.length #1 + #1 substring$ "" =
    {""}
    {
      's :=
      right.length
      {duplicate$ duplicate$ s swap$ #1 substring$ "/" = not and}
        {#1 -}
      while$
      duplicate$ #2 <
        {
          pop$ "    " s * ""
        }
        {
          duplicate$ s swap$ #1 swap$ substring$
          swap$ #1 +
          s swap$ global.max$ substring$
        }
      if$
    }
  if$
}
FUNCTION{split.name}
{
  'right.length :=
  duplicate$ right.length #1 + #1 substring$ "" =
    {""}
    {
      's :=
      right.length
      {duplicate$ duplicate$ s swap$ #5 substring$ " and " = not and}
        {#1 -}
      while$
      duplicate$ #2 <
        {
          pop$ "  " s * ""
        }
        {
          #4 + duplicate$ s swap$ #1 swap$ substring$
          swap$
          s swap$ global.max$ substring$
        }
      if$
    }
  if$
}
⟨/export⟩
```

2.4.5 Exporting fields

Here, we have four exporting functions, since we also have to deal with abbreviations:

field.export
abbrv.export
name.export
url.export

```
⟨*export⟩
FUNCTION{field.export}
{
  duplicate$ missing$
    'skip$
    {
      left.delim swap$ * right.delim *
      swap$
      "  " swap$ * " = " * left.width space.complete
      swap$ "," *
      {duplicate$ "" = not}
        {
          right.width split.string 't :=
          *
          write$ newline$
          "" left.width space.complete t
        }
      while$
    }
  if$
  pop$ pop$
}
```

```
FUNCTION{abbrv.export}
{
  duplicate$ missing$
    'skip$
    {
      swap$
      "  " swap$ * " = " * left.width space.complete
      swap$ "," *
      {duplicate$ "" = not}
        {
          right.width split.string 't :=
          *
          write$ newline$
          "" left.width space.complete t
        }
      while$
    }
  if$
  pop$ pop$
}
FUNCTION{name.export}
{
  duplicate$ missing$
    'skip$
    {
      left.delim swap$ * right.delim *
      swap$
      "  " swap$ * " = " * left.width space.complete
      swap$ "," *
      {duplicate$ "" = not}
        {
          right.width split.name 't :=
          *
          write$ newline$
          "" left.width space.complete t
        }
      while$
    }
  if$
  pop$ pop$
}
FUNCTION{url.export}
{
  duplicate$ missing$
    'skip$
    {
      left.delim swap$ * right.delim *
      swap$
      "  " swap$ * " = " * left.width space.complete
      swap$ "," *
      {duplicate$ "" = not}
        {
          url.right.width split.url 't :=
          *
          write$ newline$
          "" left.width space.complete t
        }
      while$
    }
  if$
  pop$ pop$
}
⟨/export⟩
```

2.4.6 Handling abbreviations

Abbreviations are difficult to deal with if we wish to still use them, since BibTEXwill expand them before we can do anything. All we can do is to define them in a special way, in order to be able to get back to the abbreviations later on. This is precisely what we do:

jan-dec
acmcs-tcs
remove.exports.from.months
remove.export.from.journal

```
⟨*export⟩
MACRO{jan}{"export-jan"}
MACRO{feb}{"export-feb"}
MACRO{mar}{"export-mar"}
MACRO{apr}{"export-apr"}
MACRO{may}{"export-may"}
MACRO{jun}{"export-jun"}
MACRO{jul}{"export-jul"}
MACRO{aug}{"export-aug"}
MACRO{sep}{"export-sep"}
MACRO{oct}{"export-oct"}
MACRO{nov}{"export-nov"}
MACRO{dec}{"export-dec"}
MACRO{acmcs}{"export-acmcs"}
MACRO{acta}{"export-acta"}
MACRO{cacm}{"export-cacm"}
MACRO{ibmjrd}{"export-ibmjrd"}
MACRO{ibmsj}{"export-ibmsj"}
MACRO{ieeese}{"export-ieeese"}
MACRO{ieeetc}{"export-ieeetc"}
MACRO{ieeetcad}{"export-ieeetcad"}
MACRO{ipl}{"export-ipl"}
MACRO{jacm}{"export-jacm"}
MACRO{jcss}{"export-jcss"}
MACRO{scp}{"export-scp"}
MACRO{sicomp}{"export-sicomp"}
MACRO{tocs}{"export-tocs"}
MACRO{tods}{"export-tods"}
```

```
MACRO{tog}{"export-tog"}
MACRO{toms}{"export-toms"}
MACRO{toois}{"export-poois"}
MACRO{toplas}{"export-toplas"}
MACRO{tcs}{"export-tcs"}
INTEGERS{ intxt }
FUNCTION{remove.exports.from.months}
{
  #0 'intxt :=
  duplicate$ missing$
    'skip$
    {'t :=
    ""
    {t #1 #1 substring$ "" = not}
      {
      t #1 #7 substring$ "export-" =
        {intxt
           {right.delim * #0 'intxt :=}
           'skip$
         if$
         duplicate$ "" =
           'skip$
           {" # " *}
         if$
         t #8 #3 substring$ *
         t #11 global.max$ substring$ 't :=}
        {intxt
           'skip$
           {duplicate$ "" =
              {}
              {" # " *}
            if$
            left.delim * #1 'intxt :=}
         if$
         t #1 #1 substring$ *
         t #2 global.max$ substring$ 't :=}
      if$
      }
    while$
    intxt
      {right.delim *}
      'skip$
    if$
    }
  if$
}
FUNCTION{remove.export.from.journals}
{
  duplicate$ missing$
    'skip$
    {
      duplicate$ #1 #7 substring$ "export-" =
        {#8 global.max$ substring$}
        {left.delim swap$
         right.delim * *}
      if$
    }
  if$
}
⟨/export⟩
```

2.4.7 Now, we export...

We gather everything. This is were special fields must be added for being exported:

entry.export.standard
entry.export.extra
entry.export
export

```
⟨*export⟩
FUNCTION{entry.export.standard}
{
  "address" address field.export
  "author" author name.export
  "booktitle" booktitle field.export
  "chapter" chapter field.export
  "crossref" crossref field.export
  "edition" edition field.export
  "editor" editor name.export
  "howpublished" howpublished field.export
  "institution" institution field.export
  "journal" journal remove.export.from.journals abbrv.export
  "key" key field.export
  "month" month remove.exports.from.months abbrv.export
  "note" note field.export
  "number" number field.export
  "organization" organization field.export
  "pages" pages field.export
  "publisher" publisher field.export
  "school" school field.export
  "series" series field.export
  "type" type field.export
  "title" title field.export
  "volume" volume field.export
  "year" year field.export
}
FUNCTION{entry.export.extra}
{
  "abstract" abstract field.export
  "doi" doi field.export
  "eid" eid field.export
  "isbn" isbn field.export
  "issn" issn field.export
  "language" language field.export
  "url" url url.export
}
FUNCTION{entry.export}
{
  entry.export.standard
  entry.export.extra
}
FUNCTION{export}
{
  "@" type$ * "{" * cite$ * "," * write$ newline$
  entry.export
  "}" write$ newline$ newline$
}
⟨/export⟩
```

2.4.8 Miscellanea

We also have to handle preamble, and to define functions for each entry type (we won't use them but otherwise, BibTEXwould complain).

preamble
header
entries.headers
article-unpublished

```
⟨*export⟩
FUNCTION{preamble}
{
preamble$ duplicate$ "" =
  'pop$
  {
    ",-------------------." write$ newline$
    "|     PREAMBLE      |" write$ newline$
    "`-------------------'" write$ newline$ newline$
    "@preamble{ "  swap$
    quote$ swap$ * quote$ *
    {duplicate$ "" = not}
      {
        right.long.width split.string 't :=
        *
        write$ newline$
        "" left.short.width space.complete t
      }
    while$
    "}" write$ newline$ newline$
    pop$ pop$
  }
if$
}
FUNCTION{header}
{
%"** This file has been automatically generated by bibexport **"
%write$ newline$
%"** See   http://www.lsv.ens-cachan.fr/~markey/bibla.php    **"
%write$ newline$
%"** for more informations about bibexport.                  **"
%write$ newline$
newline$
}
FUNCTION{entries.header}
{
preamble$ "" =
  'skip$
  {
    ",-------------------." write$ newline$
    "|  BIBTEX ENTRIES   |" write$ newline$
    "`-------------------'" write$ newline$ newline$
  }
if$
}
FUNCTION{article}{export}
FUNCTION{book}{export}
FUNCTION{booklet}{export}
FUNCTION{conference}{export}
FUNCTION{habthesis}{export}
FUNCTION{inbook}{export}
FUNCTION{incollection}{export}
FUNCTION{inproceedings}{export}
FUNCTION{journals}{export}
FUNCTION{manual}{export}
FUNCTION{mastersthesis}{export}
FUNCTION{misc}{export}
FUNCTION{phdthesis}{export}
FUNCTION{proceedings}{export}
FUNCTION{techreport}{export}
FUNCTION{unpublished}{export}
⟨/export⟩
```

2.4.9 Main program

We now can execute and iterate those functions:

```
⟨*export⟩
READ
EXECUTE{header}
EXECUTE{preamble}
EXECUTE{entries.header}
ITERATE{export}
⟨/export⟩
```