A Bibliographer's Toolbox

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
WWW URL: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 581 4148
Internet: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org

Abstract

This article surveys a portion of a set of software tools that I have developed over the last decade for the production, maintenance, testing, and validation of very large bibliographic archives.

It provides resource locations for all them, and shows how they can make bibliography preparation and maintenance more productive, and much more reliable.

Summary

1 Introduction

> LEXICOGRAPHER, N. A WRITER OF DICTIONARIES; A HARMLESS DRUDGE THAT BUSIES HIMSELF IN TRACING THE ORIGINAL, AND DETAILING THE SIGNIFICATION OF WORDS.
> SAMUEL JOHNSON
> *Dictionary of the English Language* (1755)

> BIBLIOGRAPHER, N. SEE LEXICOGRAPHER.
> ANONYMOUS

Large documents, especially in technical fields, often contain a list of other related documents, in the form of a bibliography or reference list. That list usually appears at the end of the document, but may instead be sprinkled through it in footnotes, or collected in endnotes, or be divided in multiple parts, with one part at the end of each major document division, such as a chapter.

Before computers came into wide use for document preparation, these reference lists were tedious to prepare, and were often sparse, with authors after the first reduced to the Latin catch-all *et al.*,[1] article titles, issue numbers, and months omitted, and page ranges reduced to the initial page.

This is a disservice to the reader, who has little idea what the referenced publication is about, and who then must work harder to find it. Much of the chemistry literature still follows this practice.

Editor's note: As an experiment, this paper is not typeset in Computer Modern. The main text is Bitstream Charter, with monospaced material in Bigelow & Holmes LuxiMono. We do not plan at this time to change from Computer Modern in general, but we still welcome your comments.

[1] A colleague once quipped: "When you see a paper cited as *Jones et al.*, it means that Jones got the credit, but Al did the work."

The tedious manual labor of preparing a reference list had to be repeated if the publication style changed, and error rates were high.

Bibliographic database systems, such as bib/refer, bibix, BibTeX, EndNote,[2] Papyrus,[3] and ProCite,[4] have made it possible to remedy this situation. Bibliographic data can now be carefully prepared once and for all, freely used by anyone, automatically reformatted into scores of styles, and converted between different database formats with reasonable ease.

As one example of a significant shared collection, the *Karlsruhe Bibliography Archive*[5] in mid-2004 contained about 1.4M references in BibTeX form, covering major areas of computer science, numerical analysis, electronic document production, fonts, and typography. A good portion of that archive is the result of my own work.[6]

A key feature of all of bibliographic database systems is the separation of *database markup* (data types author, title, journal, year, etc.) from *presentation markup*. Style files guide bibliographic software in the conversion from database form to presentation form. For example, a famous letter to the editor might be marked up like this in a BibTeX file:

```
@Article{Dijkstra:1968:GSC,
  author =       "Edsger Wybe Dijkstra",
  title =        "Go to statement considered
                 harmful",
  journal =      j-CACM,
  volume =       "11",
  number =       "3",
  pages =        "147--148",
  month =        mar,
  year =         "1968",
  CODEN =        "CACMA2",
  ISSN =         "0001-0782",
  note =         "This letter inspired scores of
                 others, published mainly
                 in SIGPLAN Notices up to
                 the mid-1980s. The best-known
                 is \cite{Knuth:1974:SPG}.",
}
```

For comparison, a subset of that data might be marked up in bib style like this:

```
%A Edsger Wybe Dijkstra
%T Go to statement considered harmful
%J Comm. ACM
%V 11
%N 3
%P 147-148
%D March 1968
```

In author-date reference style, inline citations to it might read *(Dijkstra, 1968)*, with the bibliography item in a .bbl file marked up like this for TeX or LaTeX:

```
\bibitem[\protect\citename{Dijkstra, }1968]
        {Dijkstra:1968:GSC}
Dijkstra, Edsger~Wybe. 1968.
\newblock Go to statement considered harmful.
\newblock \emph{Communications of the ACM},
\textbf{11}(3), 147--148.
\newblock This letter inspired scores of others,
published mainly in SIGPLAN Notices up to the
mid-1980s. The best-known is
\cite{Knuth:1974:SPG}.
```

It produces typeset output like this:

> Dijkstra, Edsger Wybe. 1968. Go to statement considered harmful. *Communications of the ACM*, **11**(3), 147–148. This letter inspired scores of others, published mainly in SIGPLAN Notices up to the mid-1980s. The best-known is (Knuth, 1974).

Notice in this example that it is possible for bibliographic entries to cite other entries; the cited entries are automatically retrieved and formatted by the software, without any extra effort by the document author.

A chemistry-literature citation of the same data might have a numeric superscript pointing to a footnote,[7] without any change to the citation in the document.

Bibliographic databases deserve to be widely used, freely shared, and contributed to by many. The time has come to abandon the cryptic reference-list practices of the past that were developed primarily as labor-saving devices, and replace them with accurate, and detailed, reference lists, as exemplified by the bibliography at the end of this article.

The databases that I have developed intentionally avoid unnecessary abbreviations: does *J. chem. phys.* mean *Journal of Chemical Physics*, or *Journal de chimie et physique*, or maybe even *Journal of Chemosurgery and Physiology*? They also supply details that historically were omitted, such as book and periodical numbers, library catalog numbers, and publisher states-or-provinces and countries. While there are many publishers in London, England, and in Paris, France, there are also some in

[2] http://endnote.com/

[3] http://www.researchsoftwaredesign.com/

[4] http://www.procite.com/

[5] http://liinwww.ira.uka.de/bibliography/

[6] http://www.math.utah.edu/~beebe/bibliographies.html

[7] E. Dijkstra, Comm. ACM **11** 147 (1968). [Notice here that the reader may have to search up to twelve monthly issues of this journal to find the article, because page numbers restart at one each issue, and the abbreviated reference style omits the issue and month information.]

London, Ontario, Canada, and maybe even a small one in Paris, Idaho, USA, not very far from my home in Utah.

The rapid spread of the Internet, the use of the World-Wide Web for sharing electronic documents, and free and independent search engines for finding them, requires a radical change to past practices. We need to get our old documents onto the Web, and we need to make our new documents include pointers to electronic versions of all documents that we reference.

If Brewster Kahle's magnificent vision [29, 16] of having all of the world's entire historical literary production online in the Internet Archive Wayback Machine[8] for free access by everyone comes to fruition, we will have an unprecedented resource of human knowledge. We will need to index it, search it, and be able to reference any part of it, so that others can find the same material. We also need to replicate it in many countries, and in many storage technologies, to avoid repeating the catastrophic destruction of the great Library of Alexandria, though just when, or how often, that happened is uncertain.[9]

2 The problem(s)

2.1 Data errors

The biggest problem with databases is *errors*: insurance-company studies have reported errors in as many as a third of all stored fields [31], and sometimes in up to 80% [26].

Bibliographic data is no exception; the quality of much of the Karlsruhe archive (apart from my own section) is poor, entirely due to careless preparation and checking on the part of human contributors.

The same criticism applies to many commercial and governmental bibliographic databases, such as *Compendex*, *Medline*, *Science Citation Index*, and the many *OCLC* databases. The *Uncover* database is so bad that virtually all of its data is suspect.

You may well be able to *find* a publication listed in these sources, but you cannot rely on their information to provide you with correct bibliographic data. Every field of data potentially contains errors introduced by careless, or low-paid, human typists. Mathematical markup is almost nonexistent, and when it is there, it is frequently unintelligible or completely wrong. Accents on letters are not recorded, which is particularly offensive when personal names are grossly misspelled.

Notable exceptions in database quality are the American Mathematical Society's *MathSciNet* facility,[10] the European Mathematical Society's *Zentralblatt MATH* archives,[11] and the SIAM journal publication lists.[12] These organizations deserve high praise for the care with which they have recorded bibliographic data.

Fortunately, major journal publishers now offer Web pages with issue tables of contents, and it is often possible to write software to automatically derive bibliographic entries directly from the HTML or XML markup. Since the publisher has the original data from which the bibliographic information is derived, and has a business interest in its quality, we can hope that fewer errors exist. I'll have more to say about this later.

Why does citation accuracy matter? Here are some reasons:

- Getting your references correct is not just a matter of ethical and professional responsibility: it shows respect for your reader.
- There are many more readers than writers, authors, and bibliographers. Your work will be seen by many, and for shared bibliographic data, also used and reused by many. Errors in such data will be amplified many times.
- When literature references are inaccurate, your competency is called into question. Erroneous bibliographic citations suggest carelessness in the rest of your work.
- An encouraging recent trend is for the full text of articles to be available electronically, with Web cross-links from the references to the publications that they cite. The ACM, AMS, EMS, and IEEE all have digital-library projects underway that provide such linking. Accurate bibliographic data is essential for correct links.

  The huge ArXiv e-print service[13] in physics, mathematics, non-linear science, computer science, and quantitative biology does not currently provide such links, possibly to avoid competition with professional-society and commercial collections, but live links between documents are too valuable to continue to ignore in the future.
- In business and law, contracts and legal decisions depend critically on document accuracy, including references to prior agreements and legal cases.

[8] http://www.archive.org/

[9] http://en.wikipedia.org/wiki/Library_of_Alexandria

[10] http://ams.org/mathscinet/

[11] http://www.emis.de/

[12] http://epubs.siam.org/

[13] http://www.arxiv.org/

2.2 Markup features

BibTeX goes further than most other bibliographic systems in addressing the need for adequate markup. While its name implies a connection with TeX, its design was strongly influenced by the Scribe document-formatting system, and there was a period, now long past, where database interchangeability between the two systems was essential. There are BibTeX styles available that format references for consumption by other typesetting systems, notably, nroff and troff, and some of the commercial bibliographic database systems can import and export BibTeX data.

One of the most important design decisions in BibTeX is that the publication types and the data field types are *not* hard-coded into the BibTeX program. Instead, that program knows what their syntax is, but their names are defined only in style files. All such style files recognize a standard set of document types (`Article`, `Book`, `PhDThesis`, `TechReport`, `...`), and a standard set of field names (`author`, `title`, `year`, `...`). However, database entries can contain additional field names: unknown ones are silently ignored, even though they might simply be misspelled!

This flexibility has been enormously helpful in extending the markup to handle needs that were not foreseen when BibTeX was designed. Perhaps the most notable of these is the World-Wide Web, with uniform resource locators (URLs) providing location information as a supplement, or alternative, to conventional data, like `journal`, `volume`, `pages`, `year`, and so on.

Other examples include the `DOI` (Digital Object Identifier),[14] `ISBN` (International Standard Book Number), `ISSN` (International Standard Serial Number), and `CODEN` (Chemical Abstracts serial number) field names, which provide handles that uniquely identify a document, a book or a periodical. The `is-*.bst` style files are extensions of the BibTeX base styles that recognize these field names, and several others, and also understand an additional document type, `Periodical`.

Compared to other systems, BibTeX markup is clear and simple. It requires only ordinary text files, and those files are readily understandable by anyone who can read English, even an elementary-school child. That is not the case with other systems; for example, in Unix bib, you might be able to guess that `%A` stands for *author* , but you probably have no idea what `%Q` means. [It is used for a corporate or 'foreign' author.]

The clarity of BibTeX markup has turned out to have an unexpected, but extremely valuable, benefit. Entries from BibTeX files are readily found by Web search engines: try searching for *+@Book +Knuth* in your favorite search engine.

One important problem that BibTeX has solved cleanly is the need for identification of proper nouns, so that those styles that downcase titles can be correctly supported. Thus, a field assignment like

```
  title = "The Use of {Green} Functions for
           Modeling Growth of Green Algae",
```

lets BibTeX preserve lettercase on the first instance of *Green*, and downcase the second, when the style calls for it. German capitalizes all nouns, so titles in that language need only an outer brace layer to eliminate disastrous downcasing:

```
title = "{Einschlie{\ss}en der L{\"o}sungen
         von Randwertaufgaben}. ({German})
         [{Bracketing} Solutions to Boundary
         Value Problems]",
```

The lack of such markup in other systems forces downcasing of the title data, thus losing possibly-important information about the lettercasing in the original publication.

It is certainly bad form for the database to lose information; any such data reduction should be entirely up to the style.

2.3 Markup deficiencies

While I am convinced that BibTeX markup is currently the best choice for bibliographic databases, the experience of personally creating more than a third of a million entries, and using them daily for more than a decade, has turned up limitations that must be addressed in the final, and frozen, release of BibTeX. Its author (Oren Patashnik) and I have had numerous electronic and face-to-face exchanges about these issues, and I'm confident that proper solutions will be found.

Here are just two examples of markup deficiencies:

- There is no author/editor value markup to distinguish between levels of authors. For example, my entry for the second edition of *The LaTeX Companion* [30] includes this (unused) field:

  ```
  remark = "Authors listed as: Frank
           Mittelbach and Michel Goossens
           with Johannes Braams, David
           Carlisle, and Chris Rowley, and
           with contributions by Christine
           Detig and Joachim Schrod.",
  ```

  These wonderful people have done a truly outstanding job in this new edition, and it is unfair

[14] `http://www.doi.org/`

to omit any of their names in the database. Consequently, they each appear in the author field separated by the BibTeX keyword and, so the book entry at the end of this article loses the level information.

- Parsing of personal names into *first*, *von*, and *last* parts is not general enough to handle different name order, such as Chinese, Hungarian, Japanese, and Vietnamese, where the family name comes first.

  Nor does it properly handle the case of South Indian authors with a single name: *Arvind* is a well-known example in computer science.

  It also fails for Spanish names with paternal and maternal contributions: *Juan García y Rodriguez* may be known as Juan García R., or just J. García, when the maternal part is abbreviated or dropped.

  As we learn more about the personal-name conventions of other parts of the world, more limitations of the current markup will certainly be found.

Much more could be said about markup deficiencies and challenges, but instead, I now want to turn to a review of tools that can be profitably used to ease the job of bibliography-entry preparation, checking, and typesetting.

3 Creating bibliographic data

A common way to create bibliographic entries is by manual data entry. For most systems, this is best done with the help of templates. I am not at all fond of the window-based clients, provided with most commercial systems, that present little boxes to be filled in. Their editing capabilities are badly crippled, and they provide no automated way to create parts of entries in advance, something that I routinely do in preparation of entries for journal-specific bibliographies. The little-boxes clients may be tolerable for creation of a dozen bibliographic entries, but they are completely hopeless for the creation of hundreds of thousands.

3.1 The emacs environment

The "Extensible, Customizable, Self-Documenting Display Editor", emacs, [3, 13, 14, 15, 17, 21, 28, 34, 35, 36, 37] provides the finest editing environment that I know of, far beyond that of any other editor ever developed and widely deployed. emacs celebrates its thirtieth birthday in 2006[15,16,17] and for most of its users, has changed their lives, making computers, document entry, and programming, more accessible. Its age is a badge of honor, one that it shares with TeX, two years its junior. My computer-input capability jumped tenfold when I adopted emacs more than twenty-five years ago.

What makes emacs different from many other editors arises from five fundamental design principles:

- Commonly-used editing commands are bound to easily-learned keys, but those bindings are always customizable. Most users retain the default bindings of the general commands for copying, deletion, insertion, and movement. However, key bindings of more specialized editing commands usually change with the type of data being edited.
- Time-critical editing tasks and display management are carried out by the emacs kernel (originally written in PDP-10 assembly code, now in C), but the bulk of editing functionality is handled by interpreted code (originally TECO, now Lisp). That code can be developed and tested in an interactive editing session, saved in a library file, optionally compiled for efficient reuse, and dynamically loaded into other editing sessions.
- While the emacs kernel remains under control of a handful of architects, led by Chief Architect and Head Gnu Richard Stallman, the emacs community is encouraged, and even expected, to develop specialized libraries that are freely shared with others.

  The assembly-code file, teco.mid, that forms the kernel of the original emacs ends with these comments:

  ```
  ;;; ITS TECO and EMACS should serve as a
  ;;; lesson to all of what can be achieved
  ;;; when programmers' creativity is not
  ;;; crushed by administrators whose main
  ;;; concern is stifling humor, stamping out
  ;;; all possibility of enthusiasm, and
  ;;; forbidding everything that isn't
  ;;; compulsory.
  ...
  ;;; You owe your improvements to us in
  ;;; return for what you see here.  If anyone
  ;;; asks you for a copy, make sure he gets
  ;;; in touch with the MIT AI Lab so he can
  ;;; get the latest stuff.
  ```

- All commands are accompanied by a short, but usually entirely sufficient, string of documentation that can be displayed with just a few keystrokes.

[15] http://www.gnu.org/software/emacs/emacs.html

[16] http://en.wikipedia.org/wiki/Emacs

[17] http://www.jwz.org/doc/emacs-timeline.html

- Full documentation is provided in the form of online text files, extensively cross-referenced, and linked into a tree or graph structure.

  Two keystrokes get you into the emacs info system for viewing the documentation, and two dozen single-character commands let you easily navigate through it. When you leave it to return to editing, and then subsequently reenter the info system, you are positioned exactly where you were when you left it, and the history of where you've been is intact.

  The info system contains highly-readable self-guided tutorials on info and emacs, making both systems rapidly accessible to new users. When emacs is built for a windowing system, it provides a conventional toolbar that makes it easy for novices to perform basic editing tasks without any knowledge of key bindings.

  The info system was hypertext [11] fifteen years before the idea was reinvented for the World-Wide Web, but the idea goes back to at least Vannevar Bush's far-sighted article in 1945 [12], and to work at Stanford University (the home of TeX, and much else) in 1962 by Douglas Engelbart and Ted Nelson. Engelbart is credited with the invention of the computer mouse and windows, and Nelson with coining the term *hypertext*.

When emacs was reimplemented in the mid-1980s for the GNU Project, it got an unlimited *undo* capability, which users find enormously liberating. If you make a mistake, you don't have to think about what editing commands to issue in order to recover: you just press the *undo* key until things are right. This facility is a sixth important design goal, even though it was not part of the original implementation.

TeX users will see a strong similarity between the developments of TeX and METAFONT, and the emacs editor, which is not surprising, since the Grand Wizard of TeX himself is a diehard emacs user. The emacs libraries have their analogues in the packages of the LaTeX world.

emacs has superb support for the creation of BibTeX data, and I've written more than a dozen related libraries[18] to further enhance the editing provided by the default BibTeX library. These libraries provide more than 350 specialized functions for editing BibTeX data.

For example, in an emacs session for a BibTeX file, three quick keystrokes, or more awkwardly, selection of an item from a pull-down menu, generates a template like this, with the cursor positioned before the comma on the first line, ready for data entry:

```
@Article{,
  author =       "",
  title =        "",
  journal =      "",
  year =         "",
  OPTvolume =    "",
  OPTnumber =    "",
  OPTpages =     "",
  OPTmonth =     "",
  OPTnote =      "",
  acknowledgement = ack-nhfb,
  bibdate =      "Tue Jun 29 11:54:21 2004",
}
```

A tab character moves to the next value field, and two keystrokes remove the OPT prefix, which is there to remind the typist that BibTeX considers that particular field optional.

The acknowledgement and bibdate fields are my own personal customizations. The latter holds a revision date that provides critical information for other software tools, such as bibextract.[19] They can extract entries matching specified patterns, for example, all those with 2003 and 2004 in the bibdate value, making it easy to find out what's new.

Once the entry is complete, two keystrokes generate a standard citation label, and when desired, two more keystrokes save the results in the filesystem.

If further processing of the new BibTeX entry, or a block of entries, is required, it takes only two keystrokes to mark a region, and then two more to get a prompt for a shell command to run on that marked data, optionally replacing it with the command output. I use that capability constantly in my bibliographic work.

Part of the extended BibTeX support in emacs provides commands for commonly-needed BibTeX-specific editing activities, such as moving from field to field, justifying string values, bracing words before and after the cursor, and supplying TeX accents.

The accent support provided by the BTXACCNT and LTXACCNT libraries is particularly convenient and noteworthy. Rather than your having to laboriously enter a backslash and a (sometimes) look-alike punctuation character or mnemonic letter, and possibly also braces, a single function key pressed *after* a character supplies the next accent from a list known to be valid for that character. Repeated

[18] http://www.math.utah.edu/pub/emacs/

[19] http://www.math.utah.edu/pub/bibextract/

presses of the *same* function key cycle through the list until the desired accent is located.

For example, after the letter o, repeated accent-key presses replace it successively by

```
{\"o}   {\'o}   {\.o}   {\=o}   {\H{o}}
{\^o}   {\`o}   {\b{o}} {\c{o}} {\d{o}}
{\r{o}} {\t{o}} {\u{o}} {\v{o}} {\~o}
```

The list is cyclic, so you never bump into its end, and if you go too far, the emacs *undo* key backtracks as far as you need to go.

Once you have found the needed accent, which you indicate just by pressing any key other than the accent key, the accent list for that character is then rotated to place the just-selected accent at the head of the list. It is likely that the same accent will be needed again, and a single accent keystroke will then retrieve it.

If you press the accent key after a letter with no known accents, emacs beeps and warns *`No accented letter match`* in the message area at the bottom of the screen.

Because TEX provides many more accents than any particular language needs, the accent lists, and thus, the accent-key presses, can be substantially shortened by selecting a language from a menu of a score of languages, including Faroese, Gaelic, Latin, Romaji, and Turkish. Of course, those lists are all customizable, so support for new ones can be added in just a few minutes.

For convenience, commonly-required external tools can be invoked from menus provided by the BIBTEX-TOOLS library, making it easy to find and run them without having to remember their sometimes long descriptive names. The menu with the functions and tools that I use often is shown in Figure 1.

Emacs is available on all popular desktop systems, although on the commercial ones, you may have to install it yourself. There is then no need to learn a new editor each time that you change platforms.

3.2 Converting Web data to BIBTEX

About the mid-1990s, another source of bibliographic data began to be available: the World-Wide Web. This takes primarily two forms: document texts that happen to contain references to other documents, and Web pages listing the contents of journal issues and conference proceedings.

The first is completely devoid of markup, and thus, requires considerable manual work to reconstruct bibliographic-database entries. All of the caveats that I raised earlier about errors apply!

The second, while varying from site to site, and even from month to month at the same site, often has enough HTML or XML markup that software can be written to construct rough BIBTEX entries that can be further cleaned up with a combination of manual editing, and many of the tools that are described elsewhere in this article.

Figure 1: Partial toolbar menu for BIBTEX-TOOLS.

```
update citation label table
print citation label table
bibcheck
bibparse
check-bbl
check-page-gaps
check-page-range
chkdelim
find-author-page-matches
find-braceable-initial-title-words
find-crossref-year-mismatches
find-duplicate-author-editor
find-duplicate-pages
find-german-titles
find-hyphenated-title-words
find-math-prefixes
find-missing-parbreaks
find-page-matches
find-possessive-title-words
find-superfluous-label-suffixes
...
```

In the best case, this conversion is nearly perfect, and fast: shortly after receiving a publication announcement in e-mail from the publisher or editor, I can sometimes create, validate, and install in the archive BIBTEX entries for a new journal issue in under five minutes. By contrast, manual creation of entries averages about that amount of time for just one.

Of course, software for the conversion of Web data to BIBTEX form is not simple to write, and could never be justified unless there is an expectation that multiple Web pages in the same format will be available for other journals of interest, and into the future. Fortunately, this has proved to be the case for some important publishers and databases, so I have been able to maintain coverage of about 300 journals. Some of them are only updated at long intervals, such as yearly, but others are updated as each new issue appears. Coverage is complete for 115 of these journals; the oldest of them is the *American Mathematical Monthly*, which goes back to 1894, and contains over 45,000 extensively-cross-referenced articles. [The *Monthly*’s Editor-in-Chief for many years is a member of my Department.]

The programming language that I have found most suitable for the conversion task is awk [2, 32, 33]. The awk language has a clean and simple syntax that borrows heavily from a small subset of the C language, and avoids the kitchen-sink syntax of certain scripting languages currently in vogue. Programs in awk are quite often right the first time, and very clear. Several of the bibliography tools described in this article are written in awk, and the books cited above contain numerous examples of awk programs.

Importantly, and unlike almost all other current scripting languages, awk has multiple implementations, three freely-distributable, and two commercial. Commercial Unix vendors supply snapshots of up to three of the free versions. This means that when a bug surfaces, it is a trivial matter to run your program with a different implementation: if the bug persists, it's yours; otherwise, it may be in the implementation.

awk's roots go back to 1978 [1], and it was designed by leading researchers in the field of parsing and compilation of programming languages at the Unix research group in *AT&T Bell Laboratories*. The language received a major overhaul in 1987, and is well described in a classic small book that appeared the following year [2].

A file count in my bibliography archive directories found 283 distinct awk programs ranging in length from about 10 lines to 12,900 lines. The arithmetic mean is 480 lines, but the geometric mean provides a more typical value: 180 lines. The total collection has nearly 122,000 lines of awk code. For comparison, TeX and METAFONT are each about 20,000 lines of prettyprinted Pascal code.

The Web conversion task does not, however, rely entirely on awk programs. Without exception, all Web pages are first processed by html-pretty,[20] my prettyprinter for HTML that standardizes markup and layout. This makes it *much* easier to write the journal- and publisher-specific awk programs.

Because these programs need occasional revision to adapt to the often-whimsical changes in Web-page format at publisher sites, I have not released them for general use, but there is little need to, at least before my demise, since they are only needed at the one site where the conversion to BibTeX takes place. However, they can certainly be made available to people with reasonable requests and expectations, perhaps for use in converting Web data for other journals.

There is a great deal of commonality in the processing of Web pages to create BibTeX output, so for most journals, a single master shell script of about 500 lines manages the conversion. It contains a giant case statement with customizations for each of about 150 journals and journal families, and ends with a loop over command-line arguments and a body with a Unix pipeline that looks roughly like this:

```
eval $PREHTMLFILTER |
  html-pretty |
    eval $POSTHTMLPRETTYFILTER |
      eval $PREAWKFILTER |
        gawk -f $BASENAME.awk \
             -v Filename=$f \
             -v JOURNAL=$JOURNAL \
             -v Journal=$JOURNAL |
          gawk -f HTML-entity-to-TeX.awk |
            gawk -f iso8859-1-to-TeX.awk |
              $POSTAWKFILTER
```

For readers unfamiliar with Unix pipelines, the vertical bar is called the *pipe operator*: it means that the output from the program on its left is the input to the one on its right. Pipeline data flows through memory buffers, rather than through files in permanent storage media, and all programs in the pipeline run simultaneously.

The output of this pipeline is trapped in a temporary file, and then further cleaned up like this:

```
biblabel $TMPFILE |
  citesub -f - $TMPFILE |
    bibsort |
      biborder |
        bibclean $BIBCLEANFLAGS |
          $POSTPOSTFILTER |
            $COMMENTFILTER
```

Most of the tools in these pipelines are described elsewhere in this article. However, these two code fragments make one thing clear: the Unix *small-is-beautiful* philosophy of software design [7, 8, 9, 19, 10, 20] is extremely powerful. Unlike the commercial offerings for bibliographic databases, there is no megalithic monstrosity that does 'everything'. Instead, fifteen separate programs each handle part of the task. Each does its job well, and each remains ignorant of what the others do.

3.3 Other external tools

There are several other tools that are worth noting briefly.

3.3.1 String abbreviations

One of the wise decisions that I made early on was to use standard abbreviations for journals, publishers, and publisher addresses, like this:

```
@String{j-QUEUE = "ACM Queue: Tomorrow's
                   Computing Today"}
```

[20] http://www.math.utah.edu/pub/sgml/

```
@String{pub-GNU-PRESS     = "GNU Press"}
@String{pub-GNU-PRESS:adr = "Boston, MA, USA"}
```

The strings `j-` and `pub-` are two of a small set of prefixes that divide BIBTEX abbreviations into namespaces, reducing collisions. Others include `inst-` for institutions, and `org-` for organizations.

The string abbreviations serve three important purposes:

- They supply a unique handle for the resource, for example, making it easy to find all of the articles in the database that are published in *ACM Queue*.
- They eliminate inconsistencies from use of differing abbreviations for the same resource.
- They provide a single point of redefinition of the resource name.

People are often surprised to learn that there are no 'official' abbreviations for journal names: publishers often disagree. All of my bibliographies therefore use full journal names in the string definitions, but if a user requires abbreviated names, it is a trivial matter to insert a following alternate definition.

Two programs, journal.awk and publisher.awk, filter an input BIBTEX stream, and output a new stream in which `journal`, `publisher`, and `address` values that have been found to match any of several common variations are replaced by the standardized abbreviations, and the data stream is prefixed by corresponding `@String{...}` definitions.

The preferred name for the string abbreviation is constructed by uppercasing an abbreviation from a large and reputable source and replacing runs of nonalphanumerics with hyphens. I prefer the catalogs of the *U.S. Library of Congress* and the *University of California Melvyl* for these sources. Thus, the journal *Biochemistry and Molecular Biology International* is found to have an abbreviation *Biochem. mol. biol. int.*, and is therefore given the handle `j-BIOCHEM-MOL-BIOL-INT`.

3.3.2 CODEN, DOI, and ISSN data

Once a handle for a journal name in a BIBTEX entry has been identified by journal.awk, it automatically supplies values for CODEN and ISSN fields, when it knows them: it does, for about 2500 journals. This makes it simple to retrofit those values into all BIBTEX entries for a particular journal.

Each BIBTEX entry, together with the string abbreviations, should be a complete and independent record of the document citation. It is definitely *not* sufficient to record those values just once, in the comment header of a BIBTEX file, because those values are lost when people copy entries from one BIBTEX file into another.

If a publisher chooses a predictable DOI or URL for a journal article, such as one based on the ISSN, volume, number, and initial page, it is then possible to automatically retrofit values for those fields into each BIBTEX entry for that journal. Sadly, the practice so far on the part of most publishers has been to use apparently-random numbers, or otherwise-unpredictable strings, in forming DOIs. This is a great shame, and a terrible loss of a great opportunity to make DOI assignment trivial for much of the world's existing and future periodical literature.

3.3.3 Missing brace protection

One of the more tedious tasks that a bibliographer must deal with properly is identification and bracing of proper nouns in titles.

In the Web pages of some publishers, there is sufficient markup and consistency that the conversion software can automatically supply those braces.

For others, the software has internal lists of names, like *Einstein*, *Navier-Stokes*, and *Schrödinger*, that frequently occur in titles, and are known to always be proper names. The *Green* example given earlier is one of the difficult cases where human understanding of the title is needed before protecting braces can be properly supplied. The easy cases are those words with mixed case, such as *BiCGS* and *McLeod*, which are always known to be proper nouns in need of protection; the conversion software braces them automatically. Another clue to a proper noun is its appearance as a possessive, as in *Another View of Einstein's Theory*.

These steps help to identify most of the proper nouns in titles, which is a particularly common practice in some areas of science, but there are always new names that turn up. For the volumes of bibliographic data that I deal with, visual examination of entries to find improperly-downcased title words is *not* practical.

To help solve that problem, and sharply reduce the number of instances of missing protecting braces, I wrote the check-bbl.awk program, with a companion shell-script wrapper, check-bbl.sh. The program searches the formatted bibliography file, usually BIBTEX's `.bbl` output file, but a thoughtless user might have created such a file by hand as well, looking for words that occur both in protecting braces, and entirely in lowercase. That way, it will likely spot an instance of *einstein* in a physics bibliography.

To make it much more likely that such errors will be detected, check-bbl.awk keeps an exception list that it updates on each run, recording protected names, and in what BIBTEX entry and file they were found. A typical entry in that list is a line like this:

```
{Wolfram}    Varney:1991:WBM    mathematica.bbl
```

At the time of writing, the exception list for my archives contains nearly 18,000 entries: that provides a *huge* reduction in the number of missed protecting braces.

3.3.4 Breaking up large BIBTEX files

When BIBTEX files get too big to manage comfortably, or worse, overflow internal tables in TEX or BIBTEX, they need to be subdivided.

The bibsplit[21] tool splits them into parts, separating entries into different output files according to one of several criteria: author, citation label, citation count, or year range.

I've had to do this numerous times with journal-specific bibliographies as their entry count grows. Usually, subdivision into decade-specific bibliographies suffices.

3.3.5 Database searching

For small databases, polished brute-force direct-search utilities like Unix grep and agrep[22,23] are reasonable solutions to the problem of finding an entry that you can only remember parts of.

agrep (approximate grep) deserves to be better known, and more widely used: it is capable of finding matches with data containing errors, such as transposition, truncation, or insertion. Here are some examples:

```
% echo Knuht | grep Knuth
# No output reported: there's an input typo

% echo Knuht | agrep Knuth
# No output: agrep normally works like grep

% echo Knuht | agrep -1 Knuth
Knuht

% echo Knuh | agrep -1 Knuth
Knuh

% echo Knooth | agrep -2 Knuth
Knooth
```

The numeric options allow matching of strings with, here one or two, errors. This sort of capability is important, because of the database-error problem discussed earlier.

Another valuable feature of agrep is its ability to return the complete paragraph where the match was found. Since I conventionally separate BIBTEX entries by a blank line, an entry *is* a paragraph, and agrep can thus report complete entries.

For large collections, however, direct search is far too slow, since the entire database corpus must be read.

One possible solution is provided by glimpse,[24] which is free to academic institutions, but requires a license fee for others. It functions with the help of a list of all unique words to each of which is attached a list of 256 buckets, each containing the names of 1/256 of the files being indexed. That list is created by glimpseindex, which is run at suitable intervals, such as nightly. The list is collapsed to 256 bits (32 bytes), and a nonzero bit in the list means that the word is found in at least one file in the corresponding bucket. glimpse then uses agrep to do its searching, but only for the subset of files known to contain the sought-for word or phrase.

For file collections running into the hundreds of megabytes, as mine do, even glimpse is not fast enough.

The solution that I happily use is bibsearch,[25] which is a frontend for the mg database [38]. Once the database index is loaded into memory, lookups are extremely fast: on our newest servers, bibsearch lookups take under one millisecond.

Unfortunately, mg has three deficiencies that prevent its use as a general Web search engine, something that I'd very much like to offer for my archives:

- mg lacks subfield searching, so it is impossible to restrict a search to find entries with Knuth in the title, but excluding entries where that name occurs elsewhere. Consequently, bibsearch often returns more than you really wanted.
- mg always strips suffixes, so a search for *hyperbola* also reports entries containing *hyperbolic*, once again returning more results than expected.
- mg supports shell escapes, making it quite difficult to provide search access to your system without giving away login access.

These are all soluble problems, but I lack the time to tackle them. Volunteers, anyone?

It may be that a new generation of Web search engines will provide a solution. While this article

[21] http://www.math.utah.edu/pub/bibsplit/

[22] ftp://ftp.cs.arizona.edu/agrep/agrep-2.04.tar.Z (Unix)

[23] http://www.tgries.de/agrep/337/agrep337.zip (Windows)

[24] http://www.webglimpse.org/

[25] http://www.math.utah.edu/pub/bibsearch/

was being written, I learned of the new estraier[26] engine that looks promising. It is also possible that conventional relational databases, such as the commercial DB2, Oracle, or Sybase systems, or the free postgresql or mysql programs, will have acceptable performance, but I have not yet found time to experiment with them for BIBTEX data retrieval.

3.4 XML for bibliographic data

In my keynote address at TUG 2003 [6], I discussed the BIBTEXML project at the Swiss Federal Institute of Technology (ETH) in Zürich, Switzerland, which has developed software for conversion between XML and BIBTEX markup of bibliographic data. At the time, their Web site was inaccessible. I'm pleased to report that the project is now back online, but at a new location.[27]

Although the syntax of XML is quite different from that of BIBTEX, both supply a lot of useful markup. If adequate support tools, analogous to BIBTEX and the many others described in this article, can be developed, then XML will be an important alternative for the representation of bibliographic data.

One important bibliography project that uses XML is the *Digital Bibliography and Library Project* (DBLP)[28] maintained by Michael Ley at Universität Trier, Germany, with about 500,000 entries in the field of computer science.

4 Checking bibliographic data

4.1 Spelling

A typical BIBTEX entry in my bibliography archives has about 750 characters and 20 lines. That is a lot of opportunity for fumble fingers to introduce typos, even if the typist is a near-perfect speller.

It is therefore imperative that bibliographic data be spell checked, preferably by more than one spelling checker. In my bibliographic work, I now use three such programs: traditional Unix spell, GNU ispell, and a new one that I developed for a book [33] that is in press at the time of writing this article. Its code is at the book's Web site.

The spelling checks are applied to the entire bibliographic file, not just to individual fields. That way, the extensive comment headers present in each file are also checked. Each file has its own exception dictionary to augment the spell checkers' own dictionaries, because the large numbers of proper names and technical words include many words that are absent from standard lists.

4.2 Delimiter balance

It takes an extremely careful proofreader to catch delimiter-balance errors, such as an open parenthesis followed by a long block of text without a matching close parenthesis. Humans can never do this job reliably, but a computer can. More than a decade ago, I wrote chkdelim,[29] and I apply it routinely to all updates of the bibliographic data in my archives. chkdelim has special knowledge of BIBTEX, Lisp, Scribe, TEX, and Texinfo. It also allows the user to request that checks for certain delimiters be suppressed: for example, angle brackets come in matching pairs in SGML, HTML, and XML, but almost never in mathematical documents.

4.3 Doubled words

Doubled-word errors (as occurs here: *in the* ***the*** *book*) are difficult for humans to spot. My dw program[30] finds them easily. It caught previously-unreported errors in both the *TEXbook* and the first edition of the *LATEX User Guide*, even though over 100,000 copies of the former had been sold, and its author offered monetary rewards for the first reports of errors in the book and its software.

4.4 File validation

While the ability of a bibliographic system to process the database is an essential check, it may not be sufficient, because uncited entries might receive only rudimentary parsing, or be ignored entirely. Worse, the exact syntax of bibliographic data may be uncertain, for lack of a formal description.

At TUG 1993, I presented a rigorous grammar for BIBTEX, and four tools based on it [4, 5]:

- bibparse[31]
- biblex (included with bibparse)
- bibunlex (included with bibparse)
- bibclean[32]

The first of these, bibparse, merely confirms that its input conforms to the grammar: a successful validation produces no output. I use this as an initial check of updates to the bibliographic archives before even attempting to run LATEX and BIBTEX: any error from bibparse immediately aborts the entire automated installation process.

[26] `http://estraier.sourceforge.net/`

[27] `http://bibtexml.sourceforge.net/`

[28] `http://dblp.uni-trier.de/`

[29] `http://www.math.utah.edu/pub/chkdelim/`

[30] `http://www.math.utah.edu/pub/dw/`

[31] `http://www.math.utah.edu/pub/bibparse/`

[32] `http://www.math.utah.edu/pub/bibclean/`

The second, biblex, parses the input and produces a token stream that is much easier to handle by other tools.

The third, bibunlex, reassembles a biblex token stream, possibly after filtering, reconstructing a valid BibTeX file.

The fourth, and most powerful, bibclean, has been of enormous importance in my bibliographic work. It is based on the same rigorous grammar as bibparse, but is implemented completely independently with a carefully-hand-coded parser, instead of the machine-generated parser in bibparse. That way, there are two independent checks on the validity of the syntax of BibTeX data.

bibclean normally produces a prettyprinted bibliography file in which numerous repairs and checks have been made on the data. For hundreds of thousands of examples, see the *TeX Users Group Bibliography Archive*[33] and the *BibNet Project* archive.[34] On request, however, bibclean produces the same token streams as biblex.

4.5 Field-value validation

One of the advantages of data markup is that it restricts the possible data values. For example, spreadsheets permit the user to assign a data type, such as currency, to a column, so that an attempt to store an alphabetic string there will be rejected. It may also be possible to specify a range of acceptable values, such as $[\$0, \$250]$ for a college student's weekly expenses.

bibclean can validate the data for many field names, and it does so primarily with the help of user-supplied patterns. The patterns are specially-designed for use with bibliographic data; they can, for example, check that a page number is a roman numeral, an arabic number, or an uppercase letter followed by a number. bibclean has special internal support for verifying the check digits in CODEN, ISBN, and ISSN values, and it also gets startup initializations that identify the valid ranges of ISBN values, which expand from year to year.

bibcheck[35] is another useful tool. Like lacheck and the more recent chktex, both of which look for common typographic-markup errors in LaTeX files, bibcheck applies a number of heuristic checks to BibTeX files.

These automated tools help, but they cannot tell whether a year value of 2003 should really be 2004. To solve problems like that, it is necessary to collect bibliographic data from multiple independent sources, and then merge that data, looking for discrepancies. Except for tiny bibliographies, this is far too tedious, and much too unreliable, to do by hand. Instead, a combination of tools provides a solution:

- bibclean first standardizes the format of the BibTeX data, greatly simplifying many other tools.
- biblabel[36] and its companion tool citesub, and an independent implementation in the emacs BIBTEX-LABELS[37] library, generate standardized citation labels that are unlikely to conflict with those of other entries, and importantly, that are easy for humans to predict as well.
- bibsort[38] sorts entries in a bibliography by any of a half-dozen different criteria.
- biborder[39] reorders fields within a BibTeX entry into a standard order, making the entries much easier to read.
- bibjoin[40] merges adjacent BibTeX entries that appear to describe the same publication, discarding duplicate data, choosing more detailed values over less detailed ones (e.g., in the author field, the longer of Donald E. Knuth and D. E. Knuth), and otherwise leaving the duplicate fields adjacent for manual correction.
- bibdup[41] checks for duplicate abbreviations and entries.
- For interactive location and repair of problems, the emacs function find-duplicate-label from the BIBTOOLS library[42] finds the next occurrence of consecutive entries with the same citation label. The function find-duplicate-key from the same library finds the next instance of duplicate adjacent field names in a single BibTeX entry.

4.6 Typesetting

Because bibliographic database software may do only cursory interpretation of field values, it is important to verify that the data can be processed by the document-formatting system.

This is particularly easy to do with BibTeX databases. Each of the BibTeX files in my archives is accompanied by a small LaTeX wrapper file that

[33] http://www.math.utah.edu/pub/tex/bib/index-table.html

[34] http://www.math.utah.edu/pub/bibnet/

[35] http://www.math.utah.edu/pub/bibcheck/

[36] http://www.math.utah.edu/pub/biblabel/

[37] http://www.math.utah.edu/pub/emacs/

[38] http://www.math.utah.edu/pub/bibsort/

[39] http://www.math.utah.edu/pub/biborder/

[40] http://www.math.utah.edu/pub/bibjoin/

[41] http://www.math.utah.edu/pub/bibdup/

[42] http://www.math.utah.edu/pub/emacs/

in minimal file-independent generic form looks like this:

```
\documentclass{article}
\begin{document}
    \nocite{*}
    \bibliographystyle{unsrt}
    \bibliography{\jobname}
\end{document}
```

The archives include these wrappers, and their output DVI, PostScript, and PDF files, to demonstrate that there are no show-stopping TEX syntax errors in the data, at least that which makes it into BIBTEX output. Errors in ignored fields will not be caught, but then, those fields are also unlikely to appear in the published bibliographies, precisely because they are ignored.

The wrappers that are actually used are somewhat more complex, because they also include a comprehensive title-word index, allowing readers who have retrieved only a typeset bibliography to quickly locate entries of interest. The index is important, because the average size of the typeset bibliographies is 125 pages, and they range from 2 to 939 pages.

5 Other platforms

All of the tools described here, and the more than 500 BIBTEX files, have been developed in the fantastic Unix programming environment. A project of this magnitude would have been infeasible on other operating systems.

The tools, and their daily invocations, make extensive use of pipelines, shell scripts, awk programs, filename pattern matching, and the ability to run multiple simultaneous processes. Repetitive tasks are managed by the make utility, the greatest software tool ever written (emacs comes second in that list).

Where does this leave potential users who are stuck with a different operating system? The answer is POSIX (pronounced *pahz-icks*, as in *positive*), a term coined by Richard Stallman for the *IEEE Standard Portable Operating System Interface* [22, 25, 24, 23]. Besides the formal Standard, POSIX is described in a few books [18, 27, 39]. Because POSIX is a significant standard, it has been implemented on scores of operating systems,[43] including all of those that you are likely to use. It just may not come by default with your vendor's operating-system distribution.

Apple Macintosh users need only upgrade to Mac OS X, which is a Unix system derived from FreeBSD and others, and then drag the `/Applications/Utilities/Terminal` icon onto the toolbar to make a Unix shell readily accessible.

Apple includes a recent version of emacs in `/usr/bin`, but it was unfortunately not built with X Window System support. However, Apple now makes available an X Window System package. Once it is installed, you can drag the `/Applications/XDarwin` icon onto the toolbar for ready use.

Work is underway to support both X and the native window system for emacs on Mac OS X, and there is a Web page describing the status of that work.[44] Should it become inaccessible, try a Web search for *`Emacs 21 for Mac OS X`*.

Following the instructions on that Web page, while writing this article, I successfully built and installed on a recent MacOS 10.3.2 system a working development-release of emacs for the native window system using the Apple-supplied `/usr/bin/gcc` compiler, which has extensions needed to build software that communicates with the operating system. Those extensions are not in standard gcc distributions.

For Microsoft Windows users, there are at least four packages that layer a POSIX or Unix-like environment on top of Windows, giving access to all of the tools described here. Consult the Web pages that I maintain on these topics for full details.[45,46,47] For an emulation-free native Unix environment, I highly recommend the outstanding commercial VMware system described in those Web documents. VMware makes it possible to run multiple native operating systems on Intel IA-32 hardware, with either a Windows or a GNU/Linux base operating system. At my Department, we use VMware to run FreeBSD, NetBSD, OpenBSD, Plan 9, Solaris x86, and Microsoft Windows on top of GNU/Linux.

6 Conclusions

This article has surveyed about two dozen software tools that support the creation and maintenance of large collections of bibliographic data in BIBTEX markup. Programs like emacs, bibclean, html-pretty, and others can be used productively in a Unix or POSIX environment to tackle projects whose size would be unthinkable for a single person on other

[43] `http://standards.ieee.org/regauth/posix/`

[44] `http://members.shaw.ca/akochoi-emacs/stories/obtaining-and-building.html`

[45] `http://www.math.utah.edu/~beebe/gnu-on-windows.html`

[46] `http://www.math.utah.edu/~beebe/windows-on-gnu.html`

[47] `http://www.math.utah.edu/~beebe/unix.html`

operating systems, or lacking most of these tools.

Space has not permitted descriptions of more than a hundred other specialized tools, but if you would find them useful, I could probably be encouraged to package up more of them on a Web site for free access and distribution. The many URLs for bibliographic resources at Utah cited in footnotes in this article are already conveniently accessible via links from a master file.[48]

A carpenter's toolbox contains large things, like drills, hammers, and saws, but it also has lots of small specialized items, such as a 3 mm nail countersink, that are used only occasionally, but do a single job well that few other tools can manage. I have no hesitation in drawing an analogy between that hardware toolbox, and my software *Bibliographer's Toolbox*.

References

[1] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. Awk — A pattern scanning and processing language. *Software—Practice and Experience*, 9(4):267–279, April 1979.

[2] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Addison-Wesley, Reading, MA, USA, 1988. ISBN 0-201-07981-X. x + 210 pp. LCCN QA76.73.A95 A35 1988.

[3] Larry Ayers. *GNU Emacs and XEmacs*. Prima Publishing, Roseville, CA, USA, 2001. ISBN 0-7615-2446-0. xxxv + 508 pp. Includes CD-ROM.

[4] Nelson Beebe. Bibliography prettyprinting and syntax checking. *TUGboat*, 14(3):222, October 1993.

[5] Nelson Beebe. Bibliography prettyprinting and syntax checking. *TUGboat*, 14(4):395–419, December 1993.

[6] Nelson H. F. Beebe. 25 years of TEX and METAFONT: Looking back and looking forward: TUG 2003 keynote address. *TUGboat*, 25(1), 2004. In this volume.

[7] Jon Louis Bentley. *Programming Pearls*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-10331-1. viii + 195 pp. LCCN QA76.6.B453 1986.

[8] Jon Louis Bentley. *More Programming Pearls: Confessions of a Coder*. Addison-Wesley, Reading, MA, USA, 1988. ISBN 0-201-11889-0. viii + 207 pp. LCCN QA76.6.B452 1988. US$18.75.

[9] Jon Louis Bentley. *Programming Pearls (reprinted with corrections)*. Addison-Wesley, Reading, MA, USA, 1989. ISBN 0-201-10331-1. viii + 195 pp. LCCN QA76.6.B453 1989.

[10] Jon Louis Bentley. *Programming Pearls*. Addison-Wesley, Reading, MA, USA, second edition, 2000. ISBN 0-201-65788-0. xi + 239 pp. LCCN QA76.6.B454 2000. US$24.95. This differs greatly from the first edition: both are well worth reading.

[11] B. Brown. The theory of HyperText. *WebNet Journal: Internet Technologies, Applications & Issues*, 2(1):46–51, 1999.

[12] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[13] Debra Cameron. *GNU Emacs Pocket Reference*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1999. ISBN 1-56592-496-7. iii + 58 pp. LCCN QA76.76.T49 C348 1998. US$6.95.

[14] Debra Cameron. *GNU Emacs — kurz & gut*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 2000. ISBN 3-89721-211-0. 60 pp. German translation of [13].

[15] Robert J. Chassell. *An Introduction to Programming in Emacs Lisp*. GNU Press, Boston, MA, USA, 2001. ISBN 1-882114-43-4. 320 (est.) pp. US$30.

[16] Stuart Feldman. A conversation with Brewster Kahle. *ACM Queue: Tomorrow's Computing Today*, 2(4):24, 26–30, 32–33, June 2004.

[17] Craig A. Finseth. *The Craft of Text Editing—Emacs for the Modern World*. Springer Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1991. ISBN 0-387-97616-7 (New York), 3-540-97616-7 (Berlin). xii + 220 pp. LCCN QA76.76.T49 F56 1991. Contains extensive discussion of design issues for text editors, with examples from Emacs. Appendix B gives sources of numerous Emacs implementations. Appendix D summarizes the TECO command set.

[18] Bill Gallmeister. *POSIX.4: Programming for the Real World*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, January 1995. ISBN 1-56592-074-0. xviii + 548 pp. LCCN QA76.76.O63 G34 1995. US$29.95.

[19] Mike Gancarz. *The UNIX philosophy*. Digital Press, 12 Crosby Drive, Bedford, MA 01730,

[48] http://www.math.utah.edu/pub/bibtools.html

USA, 1995. ISBN 1-55558-123-4. xix + 151 pp. LCCN QA76.76.O63 G365 1995.

[20] Mike Gancarz. *Linux and the Unix Philosophy*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 2003. ISBN 1-55558-273-7. xxvii + 220 pp. LCCN QA76.76.O63 G364 2003. US$34.99.

[21] Bob Glickstein. *Writing GNU Emacs Extensions*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1997. ISBN 1-56592-261-1. xviii + 215 pp. LCCN QA76.76.T49 G56 1997. US$29.95.

[22] IEEE. *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Base Definitions, Issue 6*. IEEE, New York, NY, USA, 2001. ISBN 1-85912-247-7 (UK), 1-931624-07-0 (US), 0-7381-3047-8 (print), 0-7381-3010-9 (PDF), 0-7381-3129-6 (CD-ROM). xliv + 448 pp. Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6.

[23] IEEE. *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Rationale (Informative)*. IEEE, New York, NY, USA, 2001. xxxiv + 310 pp. Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6.

[24] IEEE. *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Shell and Utilities, Issue 6*. IEEE, New York, NY, USA, 2001. xxxii + 1090 pp. Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6.

[25] IEEE. *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) System Interfaces, Issue 6*. IEEE, New York, NY, USA, 2001. xxx + 1690 pp. Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6.

[26] Dave Lenckus. Data integrity problem is creating converts: Several trends cause a switch to new systems. Technical report, Business Insurance, December 1997. From the text: "Mr. Dorn estimates there is an 8% to 20% error rate in data, spiking as high as 80% in some cases.".

[27] Donald A. Lewine. *POSIX programmer's guide: writing portable UNIX programs with the POSIX.1 standard*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1991. ISBN 0-937175-73-0. xxvii + 607 pp. LCCN QA76.76.O63 L487 1991b. US$34.95. March 1994 printing with corrections, updates, and December 1991 Appendix G.

[28] Bil Lewis, Dan LaLiberte, Richard Stallman, and the GNU Manual Group. *GNU Emacs Lisp Reference Manual, for Emacs Version 21*. Free Software Foundation, 59 Temple Place Suite 330, Boston, MA 02111-1307, USA. Phone: 617-542-5942, 2000. ISBN 1-882114-73-6. 974 pp. Two volumes.

[29] Paul Marks. Way back when: Interview with Brewster Kahle. *New Scientist*, 176(2370):46–48, November 2002.

[30] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, Christine Detig, and Joachim Schrod. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, second edition, 2004. ISBN 0-201-36299-6. xxvii + 1090 pp. LCCN Z253.4.L38 G66 2004. US$59.99, CAN$86.99.

[31] Office of Program Policy Analysis and Government Accountability. License plate seizure program's error rate still high; program should be abolished. OPPAGA Program Review 00-25, Florida State Legislature, Tallahassee, FL, USA, December 2000. 4 pp. From the abstract: "A department study conducted in October 2000 determined that the error rate for seized license plates was 34.8%.".

[32] Arnold Robbins. *Effective AWK Programming*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, third edition, 2001. ISBN 0-596-00070-7. xxiv + 421 pp. LCCN QA76.73.A95 R63 2001. US$39.95.

[33] Arnold Robbins and Nelson H. F. Beebe. *Learning Shell Scripting*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 2004. ca. 512 pp.

[34] Dominique Rodriguez. *L'essentiel de LaTeX et GNU-Emacs: manuel de réalisation de documents scientifiques, CD-ROM TeXlive'4 GNU-Emacs 20.5 pour Windows, exercices corrigés*. Informatiques. Série Réseaux et télécoms. Dunod, Paris, France, 2000. ISBN 2-10-004814-7. xv + 352 pp. Includes CD-ROM.

[35] Michael A. Schoonover, John S. Bowie, and William R. (William Robert) Arnold. *GNU Emacs: UNIX text editing and programming*. Hewlett-Packard Press series. Addison-Wesley,

Reading, MA, USA, 1992. ISBN 0-201-56345-2. xxvii + 609 pp. LCCN QA76.76.T49 S36.

[36] Richard M. Stallman. EMACS: The extensible, customizable, self-documenting display editor. In *Interactive Programming Environments*, pages 300–325. McGraw-Hill, New York, NY, USA, 1984. ISBN 0-07-003885-6. LCCN QA76.6.I5251 1984. US$34.95.

[37] Richard M. Stallman. *GNU Emacs Manual*. GNU Press, Boston, MA, USA, fifteenth edition, 2002. ISBN 1-882114-85-X. 644 (est.) pp. US$45.00.

[38] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, second edition, 1999. ISBN 1-55860-570-3. xxxi + 519 pp. LCCN TA1637.W58 1994. US$54.95.

[39] Fred Zlotnick. *The POSIX.1 standard: a programmer's guide*. Benjamin/Cummings Pub. Co., Redwood City, CA, USA, 1991. ISBN 0-8053-9605-5. xi + 379 pp. LCCN QA76.76.063 Z57 1991.