Software tools for indexing: what we need

Nancy C. Mulvany

PC-based software for document processing has become quite sophisticated, but the software tools available for embedding index entries in text files do not meet the needs of professional indexers. The author outlines basic index requirements of American publishers, and deficiencies in the current software and their user interfaces. The indexing capabilities of Microsoft Word, WordPerfect, WordStar, XyWrite, and Macrex are reviewed. Suggestions are offered for improvement in sorting algorithms, formatting capabilities, and user interface design.

There is no doubt that software (and hardware) tools for document processing on personal computers have advanced tremendously in the past five years. One indicator of the increasing development and enhancement of document processing software tools can be found in comparing 'roundup' reviews of word processors in the September 1984 and February 1988 issues of PC Magazine. In 1984 PC Magazine asked 10 writers to review 19 products. By 1988, 48 writers were needed to review 55 products. In 1984 the reviewers tested 75 different functions for each product; in 1988 over 230 functions were tested.

Mass-market word processing software now offers computer users the ability to choose fonts and leading, integrate graphics into a document, automatically produce a table of contents, check spelling, preview the page layout, and produce camera-ready book copy. Page design products, such as Ventura Publisher (Xerox) and PageMaker (Aldus), offer even the most casual user great control over the placement of text and graphics on the printed page.

There is, however, a missing link in the chain of tools available for document production. That missing link is the ability to produce an index that conforms to even the most basic indexing standards in American publishing. When the production of book-length documents is considered, the inability to produce a proper index is glaring and unacceptable.

Since virtually every major PC-based word processing product claims that an indexing module is included, it will be advantageous firstly to take note of standards in American indexing, and secondly, to review what the indexing modules in major word processing programs are in fact capable of producing.

Indexing standards in the American publishing industry

Index writing is thought by some to be tedious, mundane, thoughtless work: the simple-minded extraction of terms from a text, the assignment of a page-number, and the organization of terms into an alphabetic list. While indexing can indeed be tedious, the writing of a proper index is far from thoughtless. A proper index, i.e., one that includes selectivity and judgment, is deemed a creative, authored work by the U.S. Copyright Office, and is entitled to copyright registration and protection.

In the United States there is an ANSI standard for indexing. This standard will soon be up for review and will surely be revised to conform more closely to the British Standard 3700: 1988 and the emerging revision of the international standard, ISO 999.

In American publishing, the indexing guidelines of the University of Chicago Press are by far the most widely accepted 'standard'. While many publishers have their own guidelines for indexing, most of these are simply variations on Section 18 of The Chicago manual of style. In all fairness, however, some publishers differ from Chicago quite dramatically. For example, Stanford University Press requires that subentries be placed in page-number order rather than in an alphabetic order.

For evaluating index generation tools, a minimum requirement would be the ability to produce an index that conforms to the University of Chicago Press guidelines. While Section 18 of the Manual of style provides one of the more lengthy discussions of indexing guidelines, we can extract a few requirements that will meet the needs of many types of books. Below I have outlined the basic 'Chicago' requirements along with my own explanation of them.

Paper first presented at ICEBOL4, the Fourth International Conference on Symbolic and Logical Computing, South Dakota, 5 October, 1989

The Indexer Vol. 17 No. 2 October 1990
1. Sorting, in general

Chicago prefers letter-by-letter alphabetization (18.94). However, the Press will accept word-by-word alphabetization when it is the preferred sort order for a particular field (e.g., medical and legal indexes are often sorted word-by-word).

You will notice that no mention is made here of ASCII sort order. There is no mention made of ASCII sort order in any indexing standard from any nation! It is not an acceptable sorting order for letters of the alphabet.

Chicago describes the two alphabetizing methods in this way:

In the letter-by-letter mode one alphabetizes up to the first mark of punctuation; that is, one ignores word spaces and alphabetizes up to the comma, colon, or period at the end of the heading, or to the comma after the first part of an inverted heading. In the word-by-word mode one applies the principle through the end of the first word and then stops, using the second and subsequent words only when two or more headings begin with the same word(s). (18.92)

The advantage of letter-by-letter sorting is that most people are familiar with this method, having used directories and encyclopedias. In the indexing of computer-related material, the letter-by-letter method has the additional advantage of gathering various spellings of terms in the same place. For example, it makes little difference if a reader spells the following term in various ways: online, on line, and on-line. The entry for ‘online’ in a letter-by-letter sort will fall in essentially the same place in the index. Table 2 (below) illustrates this point (see the entries for ‘data . . .’, ‘form . . .’, and ‘online . . .’).

2. Sorting of subentries

Subentries are usually sorted in the same order as main entries; however, leading prepositions, articles, and conjunctions are ignored in the sort (18.100). A quick look at Table 2 indicates that this basic rule is broken by all word processors tested.

The reason for this rule is to pull the important concept forward for sorting purposes. This way the reader does not have to guess which of many alternate prepositions the indexer may have chosen to precede the entry. This, of course, relates to quick access to information. The less the user of an index has to figure out about index design, the quicker information can be procured.

3. Cross-references

It is hard to imagine an index that will not benefit from the inclusion of cross-references (see and see also). Cross-references provide vocabulary control and additional guidance to readers. All programs tested allowed for the inclusion of cross-references.

<table>
<thead>
<tr>
<th></th>
<th>Word</th>
<th>WordPerfect</th>
<th>WordStar</th>
<th>XyWrite</th>
<th>Macrex</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size (pages)</td>
<td>115</td>
<td>89</td>
<td>113</td>
<td>140</td>
<td>n/a</td>
</tr>
<tr>
<td>File size (Kbytes)</td>
<td>309</td>
<td>274</td>
<td>287</td>
<td>287</td>
<td>n/a</td>
</tr>
<tr>
<td>Sort &amp; format time</td>
<td>5 sec</td>
<td>37 sec</td>
<td>28 sec</td>
<td>86 sec</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>Keystrokes*</td>
<td>8</td>
<td>4 or more</td>
<td>4</td>
<td>5 or more</td>
<td>1</td>
</tr>
<tr>
<td>Number of subentry levels</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Combine page numbers?</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cross-references?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Choose sort orders?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Somewhat</td>
<td>Yes</td>
</tr>
<tr>
<td>Hide character from sort?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Force a sort order?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sort leading articles, prepositions, etc. in subentries?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Copy entries?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Auto-invert entries?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Indented style?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Run-on style?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Number of Test Entries = 27

All tests were conducted on an AT&T 386 16Mhz system with the text files stored on a RAM disk. Variations in file sizes and number of pages are due to the default formatting features of the word processors.

*Numbers of keystrokes required to embed an entry in the text in addition to the typing of the entry itself.

Table 1  Performance of index modules in word-processing programs and a dedicated indexing program (Macrex)
SOFTWARE TOOLS FOR INDEXING

4. Page number concatenation

When an entry appears on several pages in succession, it is common to express the page numbers as a page range (e.g., 5-10) rather than as a string of individual page numbers. Not only does this concatenation save precious space in an index, it is also easier for readers to use and indicates a continuous discussion of a topic. Conversely, the indexer should be able to distinguish a scattered discussion of a topic over a series of pages by NOT concatenating the page numbers. Only WordPerfect concatenated page numbers.

5. Number of subentry levels

Chicago does not specify a preferred number of subentry levels for an index because this is so dependent upon the material being indexed. Many simple indexes need only a main entry and one subentry level. When the text becomes more complex, quite often more than one subentry level will be required. For example, in legal indexing it is not uncommon to find indexes with four and five subentry levels. Since there is no fixed rule on this, indexing software tools will ideally allow for several levels of subentries, with a minimum of two subentry levels (main entry, subentry, and sub-subentry).

6. Format: run-on or indented

The University of Chicago is a bit vague on this issue. While the index to their Manual of style is presented in an indented format, many of the other books published by the Press are formatted in a run-on style. The run-on style index, also referred to as a paragraph style, saves space. The indented style, also called outline style, takes up more space, but is easier for users to scan and allows for clear, multiple levels of subentries, which are often needed in technical material.

So often space is a precious commodity in a book. Few publishers will add another signature (of pages) to accommodate a lengthy index. Thus, when one must decide between cutting entries from an index and producing the index in run-on format to accommodate the entries, obviously the decision should be to go with the run-on style.

<table>
<thead>
<tr>
<th>Word 5</th>
<th>WORDPERFECT 5</th>
<th>WORDSTAR 5</th>
<th>XYWRITE III</th>
<th>MACREX 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Of Mice and Men', 40</td>
<td>'Of Mice and Men' 40</td>
<td>Data base, 65</td>
<td>'Of Mice and Men', 40</td>
<td>data base, 65</td>
</tr>
<tr>
<td>data base, 65</td>
<td>data base 65</td>
<td>Data entry, 60</td>
<td>data base, 65</td>
<td>database, 75</td>
</tr>
<tr>
<td>data entry, 60</td>
<td>data entry 60</td>
<td>Data structure, 70</td>
<td>data entry, 60</td>
<td>data structure, 70</td>
</tr>
<tr>
<td>data structure, 70</td>
<td>data structure 70</td>
<td>Database, 75</td>
<td>data structure, 70</td>
<td>database, 75</td>
</tr>
<tr>
<td>database, 75</td>
<td>database 75</td>
<td>Features, 10, 11, 12 Features and protocols, 31 memory usage and, 33 of host mode, 32</td>
<td>features, 10-12 Features and protocols, 31 memory usage and, 33 of host mode, 32</td>
<td>features, 10-12 of host mode, 32</td>
</tr>
<tr>
<td>features, 10, 11, 12 and protocols, 31 memory usage and, 33 of host mode, 32</td>
<td>features 10-12 and protocols 31 memory usage and 33 of host mode 32</td>
<td>memory usage and, 33 of host mode, 32</td>
<td>and protocols, 31 memory usage and, 33 of host mode, 32</td>
<td></td>
</tr>
<tr>
<td>form, 80</td>
<td>form 80</td>
<td>Form, 80</td>
<td>form, 80</td>
<td>format, 80</td>
</tr>
<tr>
<td>form of input, 85</td>
<td>form of input 85</td>
<td>Form of input, 85</td>
<td>Form of input, 85</td>
<td>Form of input, 85</td>
</tr>
<tr>
<td>format, 90</td>
<td>format 90</td>
<td>FORMAT statement, 95 FORMAT statement, 95</td>
<td>FORMAT statement, 95</td>
<td>FORMAT statement, 95</td>
</tr>
<tr>
<td>FORMAT statement, 95</td>
<td>FORMAT statement 95</td>
<td>Intel 80386, 25</td>
<td>Intel 80386, 25</td>
<td>Intel 80386, 25</td>
</tr>
<tr>
<td>Intel 8088, 20</td>
<td>Intel 8088 20</td>
<td>Müller Bridge, 100 Müller Bridge, 100</td>
<td>Müller Bridge, 100</td>
<td>Müller Bridge, 100</td>
</tr>
<tr>
<td>Müller Bridge, 100</td>
<td>Müller Bridge 34</td>
<td>Mzarak Turnpike, 104 Mzarak Turnpike, 104</td>
<td>Mzarak Turnpike, 104</td>
<td>Mzarak Turnpike, 104</td>
</tr>
<tr>
<td>Mzarak Turnpike, 105</td>
<td>Mzarak Turnpike 39</td>
<td>Of Mice and Men, 30, 35 Of Mice and Men, 30, 35</td>
<td>Of Mice and Men, 30, 35</td>
<td>OF MICE AND MEN, 35</td>
</tr>
<tr>
<td>Of Mice and Men, 30, 35 on line, 22 on line, 22</td>
<td>Of Mice and Men 30, 35 on line 22</td>
<td>On line, 22</td>
<td>Of Mice and Men, 30, 35 on line, 22</td>
<td>Of Mice and Men, 30</td>
</tr>
<tr>
<td>on-line, 24 on-line, 24</td>
<td>on-line 24 on-line 24</td>
<td>Oneliners, 26</td>
<td>on-line, 24</td>
<td>on-line, 24</td>
</tr>
<tr>
<td>oneliners, 26 onion skin paper, 25</td>
<td>oneliners 26 onion skin paper 25</td>
<td>Onion skin paper, 25</td>
<td>oneliners 26 onion skin paper 25</td>
<td>onion skin paper 25</td>
</tr>
<tr>
<td>onion skin paper, 25 online, 23 online, 23</td>
<td>onion skin paper 25 online 23</td>
<td>Online, 23</td>
<td>onion skin paper 25 online 23</td>
<td>on-line, 24</td>
</tr>
<tr>
<td>online, 23</td>
<td>online 23</td>
<td>On-line, 24</td>
<td>online, 23</td>
<td>online 23</td>
</tr>
<tr>
<td>sleep therapy, 55 sleep therapy 55</td>
<td>sleep therapy 55</td>
<td>Sleep therapy, 55</td>
<td>sleep therapy, 55</td>
<td>sleep therapy, 55</td>
</tr>
<tr>
<td>Sleeping Disease, 50 Sleeping disease 50</td>
<td>Sleeping disease 50</td>
<td>'Of Mice and Men', 40</td>
<td>'Of Mice and Men', 40</td>
<td>Sleep therapy, 55</td>
</tr>
</tbody>
</table>

Table 2 Results of sorting by indexing modules in word processing programs and a dedicated indexing program (Macrex)
The above discussion is admittedly very brief. There are many concerns regarding index format that can be relevant. A more thorough discussion of indexing style features as they relate to indexing software tools can be found in Linda Fetters’ *A Guide to indexing software.*

Concordance generation

It is necessary briefly to address the so-called ‘automatic’ indexing features found in many products on the market. There is nothing automatic about proper indexing. ‘Automatic indexing’ modules do not produce an index; they instead produce a concordance, a list of the words that appear in a document, usually arranged in alphabetical order. These ‘automatic index’ generators are indeed thoughtless string parsers. The list they produce is sorted (usually in the unacceptable ASCII order), with terms followed by strings of undifferentiated page numbers. Such lists are not copyrightable. The concordance list does not satisfy the basic criteria of an index: there is no analysis of text, no gathering together of related information, and no ability to produce subentry levels.

Testing of index modules in word processing programs

The following word processing programs for the IBM PC platform were tested: MICROSOFT WORD version 5.0, WORDPERFECT version 5.0, WORDSTAR version 5.5, and XYQUEST XYWRITE III PLUS version 3.5. For comparative purposes, a dedicated indexing program, MACREX 4.01, was used to sort the test entries.

PROCEDURES

The same ASCII text file was imported into all word processing programs. A small number of index entries (27) was embedded in each of the files. For the most part, the entries were spread evenly throughout each file.

While all the programs allow users to place the cursor on a term to automatically select it as an entry, in these tests every entry was manually inserted in the text. The reason for choosing this procedure is simple: most index entries do not appear verbatim in the text. It is always necessary to manually place entries within the text files. After all the entries were embedded in the sample text files, the index generator was run.

RESULTS

The results are summarized in two tables that follow. The most disturbing aspect of the results is the lack of control over the sorting of entries and formatting of the index. The dedicated indexing program, MACREX, was set up to sort in letter-by-letter order, as specified by Chicago. Compare the MACREX column with the other columns in Table 2. Only the MACREX list is sorted correctly in letter-by-letter style.

Since the word-processing programs sort on the space character (ASCII 32), the placement of the ‘data base’ and ‘database’ entries is different. Of the four word processors, only XYWRITE offers any control over the sorting process, and that is limited at best. XYWRITE users can redefine a one character sort table. By redefining the space character (ASCII 32), XYWRITE can produce an index more closely approximating a letter-by-letter sort. The other three programs do not allow for this.

All embedded indexing programs sort on quotation marks—notice the placement of ‘Of Mice and Men’ at the top or bottom of the index. Clearly WORD and WORDPERFECT are sorting on the ASCII value (34) of the quotation marks. The default sort format for XYWRITE also produces the same results. However, since the sort table can be redefined, this curiosity could be eliminated in a XYWRITE index.

Another sorting deficiency demonstrated by all four word processors is their inability to handle the proper sorting of numbers. Note the ‘Intel’ entries. The MACREX program was set to differentiate between upper and lower case and between punctuation (this feature can be turned off when desired). The ‘Of Mice and Men’ entries in the MACREX column of Table 2 indicate the proper (and original) format of these three entries. None of the word processors distinguished the differences, which is often necessary in indexing. All four programs created two references and ignored the entry composed entirely of capital letters.

All four word processors sorted on leading prepositions and conjunctions in subentries (see the ‘features’ entry). This is incorrect. Such terms are not to be sorted. Instead, the keyword that follows the preposition or conjunction is the item to sort. Lastly, WORDPERFECT is the only program that concatenated page numbers correctly (see the ‘features’ entry).

In regard to format of the index, none of the word processors allow for formatting a run-on style index, which is often found in social science books. None of the word processors allow for explicit control over the placement of *see also* cross-references, or for annotated page references (e.g. 56(*illus.*)). There are many formatting deficiencies in these programs, far too many to enumerate. One need only look at the variety of index formats published in books to realize that is necessary to be able to produce an index in various ways.

What is needed

SORTING IMPROVEMENTS

The simplistic sorting algorithms in all the word processors are in dire need of improvement. At the very least, users should have a choice of word-by-word and letter-by-letter sorting. If the developers hope to approach the sophistication of dedicated indexing programs like MACREX and CINDEX13 far more work is in order. For instance, the programs will need to incorporate sophisticated string processing since indexes that completely follow the guidelines of the University of Chicago have more sorting requirements than simple
letter-by-letter sorting. For example, Chicago requires that names beginning with 'Mc' be sorted as though they are spelled 'Mac'. Names beginning with 'St.' are sorted as though the abbreviation is spelled out as 'Saint'.

Such peculiarities are not restricted to the University of Chicago Press; many publishers require the same type of sorting. An efficient way to accommodate these needs is to provide for a user-defined translation table. Such a table would allow users to specify that, for example, N be sorted as XYZ. Such a table would also eliminate the sorting of leading prepositions, articles, and conjunctions in subentries, which could be defined to sort as null. Another approach to the 'leading preposition' problem would be to provide a table of ignored strings. However, this would result in these strings being ignored for sorting purposes in all situations, which is not always desirable. For example, leading prepositions in main entries should be sorted, as in 'Of Mice and Men.'

Additionally, indexers often need to 'force' particular sort orders. For example, chemical names such as 'alpha-aminobutyric acid', 'beta-aminobutyric acid', and 'gamma-aminobutyric acid' are not initially sorted on the prefixes. Instead the first portion sorted is 'aminobutyric acid', after that the prefixes are sorted. All three of these entries would appear in the 'A' section of the index.

Regardless of the approach to the solution for these sorting challenges, a great deal of string processing will result. It is therefore of particular importance that the sorting algorithm be optimized to produce the quickest sort possible. This requires not only a very efficient algorithm but also an efficient use of memory.

**Formatting improvements**

Two major formatting improvements are needed. Users should be able to choose between indented or run-on format, and indentation should be user-defined; the placement of *see also* cross-references should be user-selectable. Dedicated indexing programs like Macrex offer over 100 formatting options. While it is not necessary for word-processing programs to offer all of these, there is room for a great deal of improvement.

**User interface design**

When faced with word processor indexing software, the user of dedicated indexing software is first struck by the clumsy user interface. Often indexes are written at the end of the book production cycle, when time is of the essence. Users must be able to embed index entries quickly. Previous entries must be able to be copied quickly, in order to accommodate continuous discussion of a topic. Some programs allow for a 'begin' code to be attached to an entry, later followed by an 'end' code after perhaps several pages. For true efficiency, however, embedded entries must be repeated in every paragraph contained within the continuous discussion. This would ensure that index entries are carried along with text that may be moved in the future.

Ideally, there should be an 'auto-invert' function. Often it is necessary to double-post information. Main entries become subentries and subentries become main entries in a fully inverted entry. Even more frequently, subentries are double-posted as main entries. There should be a function that accomplishes this task at the touch of a key.

A tedious part of the index editing process is checking all cross-references. Any indexing program should include cross-reference validation. Such a feature would indicate any blind or circular cross-references, e.g., 'dogs see hounds' and 'hounds see dogs'.

As the length of a document increases, the proportion of editing time to overall indexing time also increases. Therefore, in order to be efficient, users must be able to locate quickly the embedded entries in the document. Many programs 'hide' embedded entries so that the text can be read more easily. In order to locate embedded entries in these programs, it is necessary to turn on the display of command tags. This often results in a cluttered screen that is difficult to read and work with. Embedded entries could be displayed in a separate window, leaving the text itself intact except for a small marker (as in WordStar).

While writing the index the author may decide to change its structure. Thus, users of embedded indexing software should be able to execute a global search and replace command that affects only the embedded index entries rather than the text itself.

Ideally, during indexing it would be possible to review the current version of the formatted index in a window. This would reduce editing time later because the indexer could maintain vocabulary control by reviewing the terms selected earlier in the document. In order to achieve this, developers would need to implement both an auto-sort and auto-format routine.

Both the editing process and index revision process would be less time consuming if the indexer could work with a list of index entries in page-number order. Comparing such a list to the actual pages of a document aids in achieving completeness and thoroughness in index design.

Writers often complain about the time it takes to revise embedded codes after an index has been edited. For example, it is not uncommon to index a term in two forms, the singular and the plural (e.g., 'file' and 'files'). Obviously, this results in two separate index entries that must be reconciled. Many writers fix this type of problem in the index text file with an editor. When many different references are affected, it is a tedious job to return to the document file(s) and adjust the embedded entries to reflect the editing. Although it would be no small programming feat, it would be wonderful if there were a
dynamic relationship between the index text file and document text files. In other words, editing changes made in the index would be automatically made in the document file(s).

Conclusion

The current capabilities of the indexing modules of text processing software are inadequate. Even the most simple needs of the professional indexer are left woefully unfulfilled. At a point in the publishing process when users should be able to focus on the content and structure of their indexes, they are required to spend far too much time 'cleaning up' the index that is generated by these programs. Users of such programs clearly should not be forced to devote so much time to mundane and repetitive tasks that could and should be handled by the software. At the very least, these programs should sort correctly. As well, the user interface design needs to be improved so that users can quickly embed entries and later easily locate the entries for correction.

Developers should study the requirements of The Chicago manual of style and design software that meets at least these requirements. The more ambitious developers can review the features found in dedicated programs such as Macrex and Cindex. Here they will find software that has been tailored to meet the diverse needs of indexers.

Given that an index is a very important part of any book-length document, it is indeed unfortunate that current text processing software has little to offer in the realm of embedded indexing. While great strides have been made over the years in enhancing the performance of text processing software, the indexing modules have been given little attention. The types of improvements outlined in this paper are needed in order to bring the indexing modules up to par with the other features of text processing software.

References and notes

9. Interview with Harriet Oler.
10. The publishers of the word-processing programs tested are: MICR0SOFT Word (Microsoft Corp., 16011 NE 36th Way, Redmond, WA 98073-9717, 206/882-8080), WORDPERFECT (WordPerfect Corp., 288 W. Center St., Orem, UT 84057, 801/225-5000), WORDSTAR (WordStar International, 33 San Pablo Ave., San Rafael, CA 94903, 800/227-5609), XYWRITE (XyQuest Inc., 44 Manning Rd., Billerica, MA 01821, 508/671-0888).
11. A dedicated indexing program differs from the indexing modules in word processors in that it does not work with document text files. Instead, these programs are stand-alone tools that assist indexers with the indexing process. Even with the advent of sophisticated desktop publishing tools, most indexers continue to index from hard copy, i.e., final pages, page proofs. One explanation for indexing from hard copy is that the embedded indexing tools are so inefficient that it simply is not cost effective for a professional, contract indexer to use these tools.
13. CINDEX (Indexing Research, P.O. Box 27687, River Station, Rochester, NY 14627-7687, 716/461-5530).

Nancy Mulvany is the owner of Bayside Indexing Service (Kensington, CA), a provider of indexing services for technical documents. Mulvany is a specialist in the indexing of computer-related material and is currently involved in developing of hypertext systems. In addition to extensive teaching of indexing, she has been active in the American Society of Indexers for several years and is currently its president.

Plea for consistency

The wide variety of indexing terminology among databases is frustrating. The problem could be lessened by sharing vocabularies or mapping descriptors from one database to another.


The Indexer Vol. 17 No. 2 October 1990