

WORDS



A Quarterly Bulletin for Technical Writers & Communicators

Volume 2 | Issue 1 | February 2010

The beginning

- In our profession, it's up there with death and taxes: the inescapability of Microsoft Word. It's everywhere—and everywhere, every hour, someone joins the legions of those who have come to loathe its idiosyncrasies and its instability. In this issue, Geoff Hart, Fellow of the Society for Technical Communication, assures us that all hope need not be abandoned—so long as we take some simple precautions to protect ourselves from the oftentimes fathomless behaviour of MS Word.
- Many of us have had the pleasure of using Adobe FrameMaker to author our technical documents, a product whose stability and robustness provides the starkest contrast with MS Word. But this tool has its own flaws, one of which is the inconvenient default size of some of its dialog boxes. Beginning on page 5, Dave Reynolds explains how you can resize dialog boxes and the fields on them.
- For centuries, the paragraph has been an essential building block in written text. Alas, some contemporary practices—such as the omission of the first-line indent and the growing fondness for one-sentence paragraphs—is robbing writing of usability. See page 9 for reasons why these practices should be re-assessed—and abandoned.
- DITA continues to be of great interest to technical writers. Beginning on page 15, Suchitra Govindarajan explains that you don't need a lot of under-the-bonnet knowledge about DITA to start topic-based authoring according to the DITA standard.
- Our immense gratitude is owed to internationally renowned cartoonist Michael Leunig for his permission to reprint a cartoon depicting possibly the most worrying trend in modern digital communication. See page 13. Cactus, indeed.
- Finally, if you're a budding poet and have something interesting, novel or whacky to say about technical writing, you might be interested in contributing to the next issue of *Words*. See page 20 for details.

Geoffrey Marnell

Editor [geoffrey@abelard.com.au]

Contents

Protecting yourself from Microsoft Word	1
Customising FrameMaker dialog boxes.....	5
Journals.....	8
Paragraphing: a vanishing art?	9
Looking at DITA as just another tool.....	15
Book review.....	17
Miscellany	20
Mindstretchers	21

Protecting yourself from Microsoft Word ...

... with a few relevant notes for other programs

Geoff Hart

If you've used Microsoft Word for any length of time, you know that crashes and loss of work are a fact of life. Word has become much better over time, but it remains unstable and annoying to work with, particularly the Macintosh versions. But despite its many flaws, there are ways to protect yourself, and they fall into two main categories:

- perform preventative maintenance
- develop recovery strategies.

In this brief article, I'll explain some of the things you need to know to protect yourself. Although my focus is on Word 2003 and previous versions, many of the tips in this article work equally well for other word processors (*mutatis mutandis*) and other versions of Word, with appropriate modifications.

Before continuing, you need to know where to go to change certain key settings in Word:

- In Windows: **Tools > Options**
- Macintosh OS X: **Word > Preferences**. For other word processors, open the menu that has the same name as the program, and select **Preferences**.

Perform preventative maintenance

Preventative maintenance involves taking steps to prevent problems from occurring in the first place. The importance of these steps varies among versions of Word, with the steps being most necessary for older versions.

The first group of strategies involves simple housekeeping. Like most software, Word creates temporary files that it uses behind the scenes while you work. Unlike most software, Word did a generally poor job of cleaning up after itself until fairly recently in its history, making the “temporary” part of the name a notorious oxymoron.

The two most problematic types of temporary files are *work files* and *autorecover files*. Work files, which have names that end in `.tmp` or that include the words “Word work” in the file name, used to accumulate in older versions of Word. The files never got deleted on some computers, and when Word could no longer deal with all these files, it would gradually become unstable, leading to crashes and loss of information. Autorecover files are files that Word creates to let you recover as much work as possible in the event of a crash. When Word shuts down normally, rather than crashing, it erases the autorecover files as part of its shutdown routine. But when Word crashes, these files are left behind. Autorecover files are less problematic than the temporary work files, but can still cause problems if they accumulate.

The solution to instability related to the accumulation of temporary files is to learn where these files are stored, and check whether they are accumulating in sufficient quantities to pose a problem. In more recent versions of Word, the work files appear in the same directory as the document you’re working on. The autorecover files appear in a specific directory, and you can learn its location (and change that location) in this way:

1. Select **Tools > Options** (Windows) or **Word > Preferences** (Macintosh).
2. Select the **File Locations** tab.
3. In the **File types** list, click **AutoRecover files**.
4. Click the **Modify** button.

You can now use the standard Windows or Macintosh dialog box navigation features to learn the path to the current directory, to choose a new path, or even to create a new directory (perhaps one with a memorable name such as “Word autorecover files” that will make it easy to find these files in future).

To find orphaned work files that aren’t in your current working directory, close Word, then search your hard disk using the operating system’s search tool (under the Windows menu, or by pressing `COMMAND + F` in any Finder window on the Macintosh). Search for files with “Word work” in the file name, files whose name begins with the tilde character (~), or files whose name ends in `.tmp`. Depending on which operating system and which version of Word you’re using, you may need to set the options for the search function to find “invisible” files.

To find autorecover files, search for files that contain the words “autorecovery save of” in their name. Unless Word has recently crashed and you need to recover data from any of these files, simply delete them. Add a reminder to your calendar to search for these files at least monthly until you learn whether they’re accumulating on your system; if not,

you can delete this reminder, but if they are accumulating, finding and deleting them should become part of your monthly maintenance routine.

The second group of strategies involves avoiding problematic features that have never worked reliably for most

users. (Note that although these features work just fine for some users, at least for short periods, they tend to fail unpredictably for reasons nobody seems able to explain. In my books, that makes them unreliable.) These include the Fast Save function, which you can disable in the **Save** tab of the **Options** or **Preferences** dialog box. Similarly, avoid using the Versions feature (under the **File** menu), since it seems to have been implemented using the same programming code as Fast Save. The Master Documents feature should also be avoided, since it has been a frequent cause of corrupted files ever since it was introduced, and I have no evidence this feature has been fixed as of Word 2007. Finally, never embed a table in a cell of another table. Such nested tables rapidly become unstable, and can lead to crashes and lost data. If you need to divide a cell of a table into sub-cells, it’s safer to open the **Table** menu and select **Split Cells**. Specify the number of rows and columns, click **OK**, and continue working.

The third group of strategies relates to incompatibilities between versions of Word. Wherever possible, you should avoid repeatedly transferring files between versions of Word, and between versions of Word on different operating systems (for example, Macintosh versus Windows versions, North American versus Asian versions). Such transfers generally work well, with only minor incompatibilities, but any transfer that requires conversion of a document into a slightly different version of the `.doc` (Word 2003 and earlier) or `.docx` (Word 2007) file format increases your risks. I maintain four versions of Word on my computer

Key strategies

- monitor temporary files
- avoid unreliable features
- minimise version conversion

(Word X and 2008 for Macintosh, and Word 2003 and 2007 for Windows) to minimise this problem, but I'm an extreme case because I transfer files between many versions of Word on many operating systems. For most people, standardising on a single version of Word is the easiest solution.

Develop recovery strategies

Word offers two built-in protections. These can be set in the **Save** tab of the **Options** or **Preferences** dialog box (see Figure 1 below). The **Save AutoRecover info every [x minutes]** option lets you define the interval at which Word will save an autorecover file. If Word crashes, this file lets you recover some of the information that you typed since the last time you saved the file manually. On most computers and with most versions of Word, setting the interval to 10 minutes is sufficient if you're paranoid, and setting the interval to 30 minutes will be fine for most users. With these settings, Word will save autorecovery information every 10 or 30 minutes, respectively; if it crashes, you'll only lose the information you typed since the last save of the autorecovery file (a maximum of 10 or 30 minutes, respectively, of typing). If your copy of Word is particularly crash-prone, train yourself to manually save the file (CONTROL + S in Windows and COMMAND + S on the Mac) fairly frequently. Choosing to create the autorecovery files frequently seems like a logical step, but if the interval is too short and you work too long on the file, older versions of Word may run into problems with having to juggle too many files simultaneously and may crash.

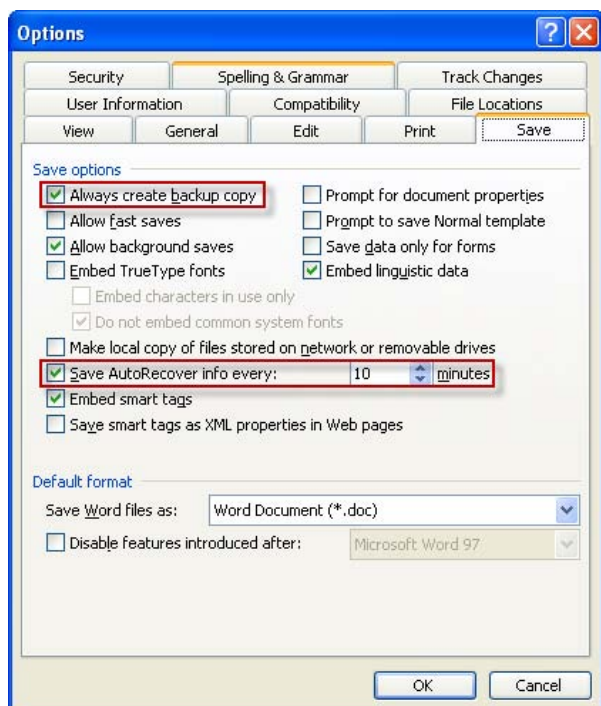


Figure 1: Recovery strategies

The second option in the **Save** tab is **Always create backup copy** (see Figure 1 above). Using this feature, Word creates a file named "Backup copy of [the name of your file]" each time you open a file. Should Word crash, or should you spend an hour working on the file and decide you prefer the original, you can simply close the open file and start working again from the backup copy. Although this is a useful feature, I've found that it's wiser to create your own backup copies. To do so, open a folder on your desktop that contains the document you're working on, and periodically make a copy (right-click the file and select **Make a copy** in Windows; select the file and press COMMAND + D on the Macintosh). Then copy it to a flash drive or e-mail yourself a copy. This can be tedious if you work on a file for long periods of time, but it places full control of backing up the files in your hands instead of relying on Word.

When a Word file becomes corrupt, it may become impossible to open it, or you may be able to open it, but find that it crashes Word. If a file is not this badly damaged, but shows signs of growing corruption—such as unusually long times to open or save the file, frequent crashes, or incorrect behaviour when you modify or update styles—consider trying this trick, first introduced by Woody Leonard in his classic book *Word 97 Annoyances*:

1. Place the cursor at the end of the file, just before the final paragraph marker.

If you can't see that marker, open the **Options** or **Preferences** dialog box and select the **Paragraph marks** checkbox in the **View** tab.
2. Press CONTROL + SHIFT + HOME (Windows) or COMMAND + SHIFT + HOME (Macintosh) to select all of the document except the final paragraph marker.
3. Copy the selected text.
4. Open a new document and paste the copied text into it.

This trick became so useful to editors that in the *copyediting-1* discussion group, it became known as "maggying" a file (after Maggie Secara, who popularised the technique).

If this trick doesn't work, you can try *inserting* the file into a new document:

1. Create a new document.
2. Select **File > Insert** and select the problem file.
3. When Word has finished inserting it, save the file under a new name.

When file corruption grows too much for these tricks to work, a new feature introduced in Word 2003 for Windows (but not, so far as I can tell, in Word 2008 for the Macintosh) can help:

1. Close the problem file and make a duplicate copy, in case you need to return to that copy.

2. Select **File > Open**. In the **Open** or **Open File** dialog box, select the problem file, but do *not* click **OK**. Instead, look for the small drop-down menu beside the **Open** button—it usually looks like a downward-pointing arrow: see Figure 2—and open that menu instead.

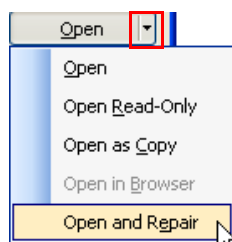


Figure 2: Open menu

3. From the drop-down menu, select **Open and Repair**.

This new feature will fix many of the most common problems that occur in Word documents.

4. If you can successfully open and repair the file in this manner, save a copy under a new name and work with that copy henceforth.

If this fails, you can sometimes open a file in other software such as OpenOffice Writer or Adobe InDesign. Both have occasionally saved my ASCII.

Trust, but not too far

I don't want to make you paranoid with this article. Word is generally a robust, stable word processor, and I spend eight or more hours working safely with it on a typical work day. That's particularly true since I switched to Word 2003, and Word 2007 is said to be even more stable. If you don't push Word quite as hard as I do, you won't often encounter grave problems. But if you do push it hard, or if you do encounter problems, the tips in this article should reduce the frequency of problems and help you recover all or most of your work should a problem arise.

Whether or not Word is giving you problems, be sure to make frequent backups, and store them somewhere safe. It's been said that there are only two types of computer users: those who have lost hours of their work, and those who soon will. My experience mentoring other writers and editors tells me this is truer than most people are willing to believe until they find themselves in the first category.

Geoff Hart

Geoff Hart is a scientific editor who specialises in working with authors for whom English is a second or third language. He is also the author of *Effective Onscreen Editing* (<http://www.geoff-hart.com/books/eoe/onscreen-book.htm>). Geoff also works as a French translator and occasionally as a technical writer. In his spare time, he commits bloggery (<http://blatherskite.dreamwidth.org/>).

Effective Onscreen Editing: new tools for an old profession

Editors are increasingly being asked to edit on the screen using a word processor, but most are finding it challenging to transfer their skills to editing with a word processor. *Effective Onscreen Editing* teaches the basics you need to learn to make the transition, plus proven tips and tricks to maximise your productivity and effectiveness. The book describes general principles valid for any software, then illustrates the principles using Microsoft Word to make them more concrete.

Available as a printed book or as an eBook optimised for onscreen reading.

Learn more at the book's Web page:
<http://www.geoff-hart.com/books/eoe/onscreen-book.htm>

 **Diskeuasis
Publishing**

In the next issue

Due out on 1st May 2010

- Documenting in an agile environment
- Introduction to DITA Conditional Processing
- Bringing language back into the spotlight
- Customising FrameMaker keyboard shortcuts
- New Book features in FrameMaker 9

The Words team

- Artwork: Christine Weaver
- Copy-editor: Marcia Bascombe
- Adviser: Mark Ward
- Editor: Geoffrey Marnell
- Contact: words@abelard.com.au

© Individual contributors or Abelard Consulting Pty Ltd, 2009
[unless otherwise noted]

Customising FrameMaker dialog boxes

Dave Reynolds

This tutorial explains how to customise FrameMaker dialog boxes to suit your particular requirements. This can be done using a program called Resource Hacker™ to edit the file that controls the appearance of each FrameMaker dialog box: `fmdlg.dll`.

The particular customisation described in this tutorial modifies the standard **Cross-Reference** dialog box so that it can display more information than it does by default. Figure 1 shows the default **Cross-Reference** dialog box and Figure 2 shows the modified version of it. Note the expanded **Paragraphs** field and the much longer list area.

The procedure outlined here can be followed for whatever dialog box you want to modify.

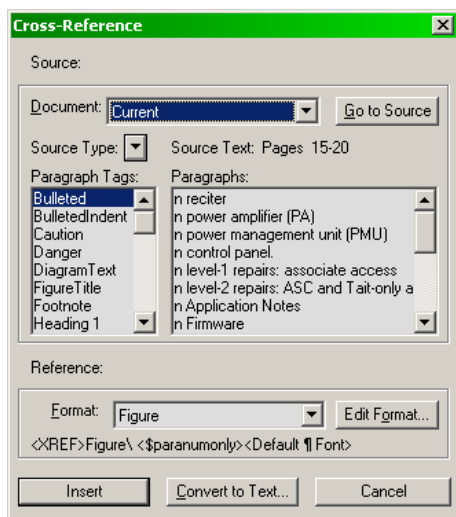


Figure 1: Default **Cross-Reference** dialog box

Constraints

I used modified dialog boxes in FrameMaker 6 for many years and found only one problem: when I opened a file with missing referenced graphics and chose to skip the missing files, FrameMaker crashed. To open a file with missing graphics, I had to revert to the original `fmdlg.dll` file.

I have used modified dialog boxes in FrameMaker 8 for a few years now without problems. There is no problem opening files with missing referenced graphics. However, I did find when testing FrameMaker 8 that I could not always use the modified FrameMaker 6 `fmdlg.dll` file. FrameMaker 8 appeared to work normally with the FrameMaker 6 file *until I tried to update a book*: and then it crashed. I had to modify the FrameMaker 8 `fmdlg.dll` file to work with books.

I have not tested modified dialog boxes in FrameMaker 7 or 9.

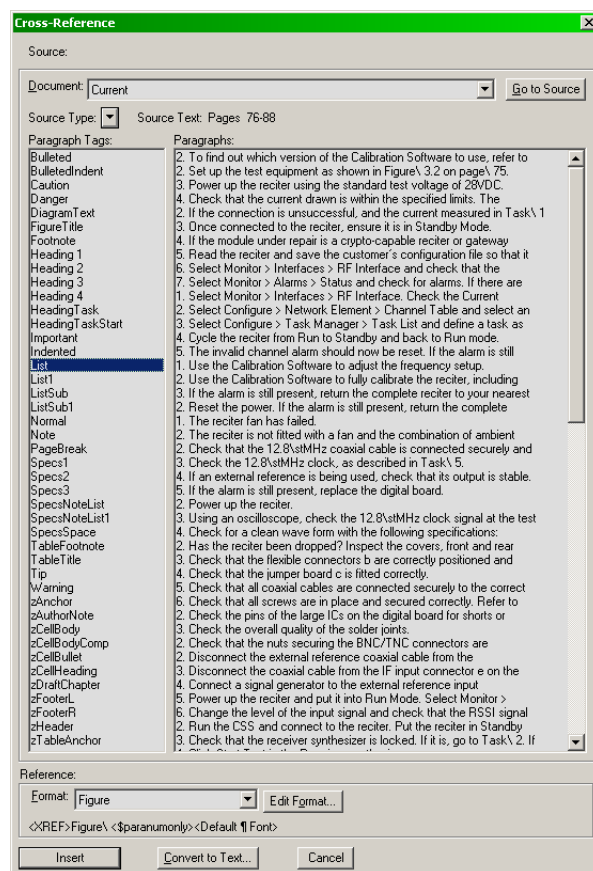


Figure 2: Customised **Cross-Reference** dialog box

Preparation

Download Resource Hacker™

Download Angus Johnson's Resource Hacker program. One of a number of download sites is:

<http://www.angusj.com/resourcehacker/>

You don't need to install the program. Just extract the files from the downloaded zip file to a suitable folder on your local drive.

Locate and copy `fmdlg.dll`

The file you need to modify is in the `fminit` folder inside your FrameMaker program folder. The default path to the `fminit` folder in FrameMaker 8 is:

```
C:\Program Files\Adobe\
FrameMaker8\fminit
```

1. Navigate to the `fminit` folder and locate `fmdlg.dll`.
2. Make a copy of `fmdlg.dll` and give it an easily recognisable name, such as `fmdlg.dll.orig`.

The copied file will enable you to revert to the default settings if necessary.

Dialog box screen shots

You may find it useful to take a screen shot of a dialog box you intend to modify. You can use this as a reference when you start your modifications. (This is especially handy if you want the modified dialog box to resemble the original dialog box but you forget the layout part-way through the exercise.)

You might also find it useful to make notes about what you've done so that you can more easily repeat the process, if necessary.

Method

Run Resource Hacker

1. Double-click `ResHacker.exe`, one of the files you extracted after you downloaded Resource Hacker.
2. Click **Run**.
3. When Resource Hacker opens, select **File > Open**.
4. Navigate to the `fminit` folder, select `fmdlg.dll` and click **Open**.
5. Expand the **Dialog** folder in the left pane. A list of the available dialog boxes is displayed, in alphabetical order (see Figure 3).

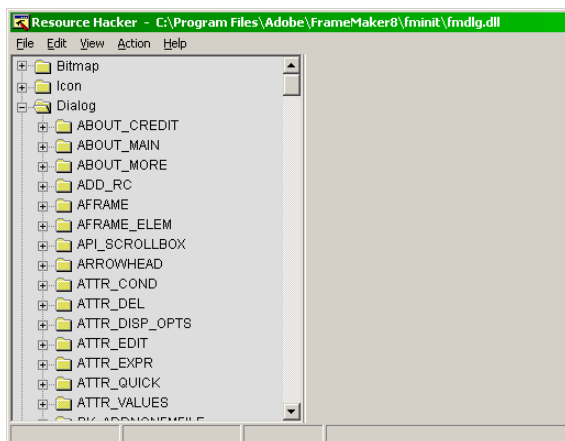


Figure 3: `fmdlg.dll` open in Resource Hacker

Edit `fmdlg.dll`

This tutorial explains how to modify the default **Cross-Reference** dialog box to show more information in the **Paragraph Tags** and **Paragraphs** fields. Expanding the **Paragraphs** field in width allows it to display more of the content of target paragraphs. This is very useful if you have many instructions that start with similar text. We will also expand the fields vertically to display more target paragraphs.

1. Scroll through the list of dialog names and double-click **XREF_MAIN**.

An option named **1033** appears as a sub-item.

2. Click on **1033**. An editable version of the **Cross-Reference** dialog box appears (see Figure 4).

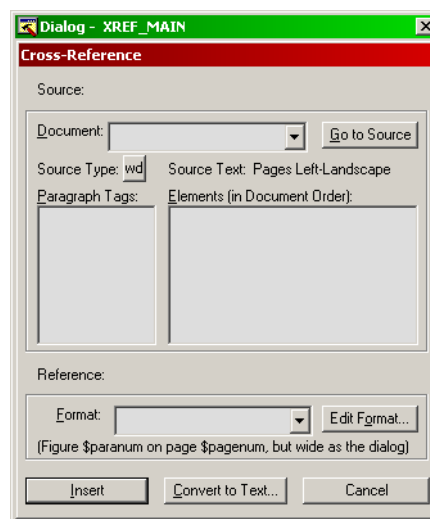


Figure 4: Default **Cross-Reference** dialog box open in Resource Hacker

Any changes you make to this window will be replicated in the **Cross-Reference** dialog box when you next open FrameMaker, so be careful what you do with this window. **!**

The code for this dialog box is displayed in the right pane of the main Resource Hacker window. Any changes made to the dialog box are recorded in this code pane. Be careful not to accidentally change or delete any of this code.

Each element in the dialog box can be selected, moved or resized. This includes text, buttons, fields and borders. Once selected, an element displays resizing handles just as in any drawing program. A red asterisk appears beside the corresponding line in the code pane.

3. Click and drag the bottom right corner to enlarge the dialog box. Note how the third and fourth "XREF_MAIN_DIALOG" coordinates at the top of the code pane change. (The expanded dialog box shown in Figure 2 has coordinates of 33, 49, 366, 484.) Moving any element in the dialog box will likewise change its coordinates in the code pane.
4. Move the **Insert**, **Convert to Text** and **Cancel** buttons to the bottom of the expanded dialog box.

You can drag them to their new position or use the arrow keys. The coordinates shown in the code pane can help you align the buttons. Note that you can only move one element at a time.
5. Move the rest of the elements under (and including) **Reference** to the bottom of the dialog box. Again, you will need to do this one element at a time. (Resort to your screen shot if you forget where each element belongs in relation to others.)

6. Enlarge the **Paragraph Tags** and **Elements (in Document Order)** fields to fill the expanded dialog box, as shown in Figure 5. (Again, the coordinates shown in the code pane can help you align the two fields.)

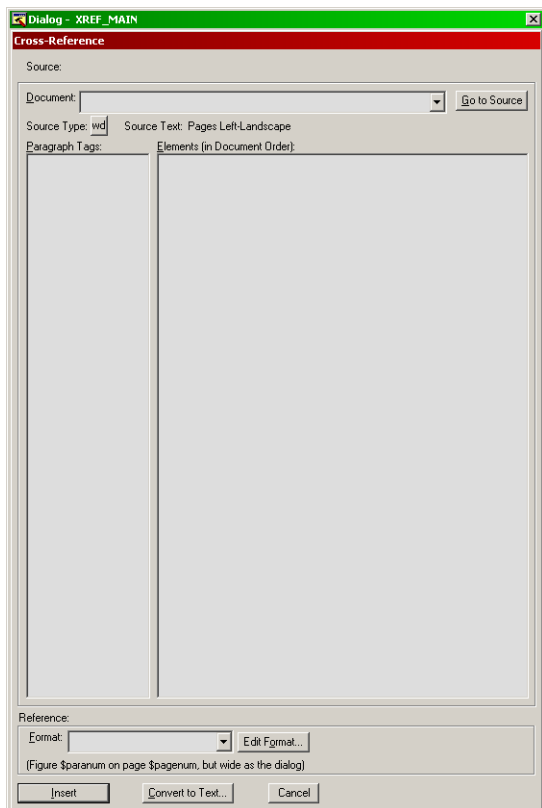


Figure 5: Modified **Cross-Reference** dialog box open in Resource Hacker

NOTE: You can't constrain the movement of elements—vertically or horizontally—with the shift or control keys as you can in a standard drawing program. If you want to move an element exactly, select it and then move it with the arrow keys. This is a lot slower, but more precise. (Alternatively, exactly specify its coordinates in the code pane.)

Note too that there is no **Undo** feature if you modify the dialog box directly (by moving and resizing its elements). The **Undo** feature is only available if you modify the code directly.

7. Once you are happy with the new layout, click **Compile Script** (above the code pane).
8. Select **File > Save**.
9. Run FrameMaker and check your new-look dialog box. If any further changes are needed, close FrameMaker and continue editing the file in Resource Hacker.

Dave Reynolds

Dave Reynolds is a senior technical author at Tait Electronics in Christchurch, New Zealand. He has been with the company since 1985, during which time he has worked on numerous documents relating to the company's two-way radio equipment.

In the next issue of *Words*, Dave explains how to customise FrameMaker shortcuts.

TRAINING COURSES

Technical, general & scientific writing, FrameMaker, Structured authoring

Abelard Consulting offers a number of courses and workshops designed to benefit technical writers, would-be technical writers, and those whose job requires them to be good communicators. The following courses are available:

- ▶ Technical Writing: An Introduction and Refresher [2 days]
- ▶ Scientific Writing: An Introduction and Refresher [2 days]
- ▶ Using Adobe FrameMaker: An Introduction and Refresher [2 days]
- ▶ Structured Authoring with Adobe FrameMaker [2 days]
- ▶ Business Writing: The Fundamentals of Clear Communication [2 days]

Where?

Scheduled classes are held in Melbourne, Sydney, Canberra, Perth, Brisbane and Adelaide. Customised on-site training is also available.

What?

To see the topics covered in a course, download a brochure from www.courses.abelard.com.au. We can also tailor a course to your specific requirements. Call 1800 601 116 for further information.

When?

Download a brochure or visit www.abelard.com.au/booking.html to see dates and locations.

By whom?

The FrameMaker and structured authoring courses are conducted by Mark Ward. Mark is a long-time user of FrameMaker and an expert in creating FrameMaker templates and structured applications.

The writing courses are designed and conducted by Dr Geoffrey Marnell. Geoffrey is the founder and principal consultant of Abelard Consulting. He also teaches Technical Writing and Editing in the English Department at the University of Melbourne and is accredited by IPED (Institute of Professional Editors). Geoffrey has more than 20 years experience as a technical writer, documentation consultant and educator.

Discounts?

Early-bird, concessional and student discounts apply. Further, on-site training is charged at a flat rate regardless of the number of participants.

ABELARD consulting Pty Ltd

Freephone: 1800 601 116

Telephone: +61 3 9596 3456

Email: courses@abelard.com.au

Web: courses.abelard.com.au

Journals

Rhetoric, Professional Communication, and Globalization

Barry Thatcher, Kirk St. Amant

This journal publishes research articles on the theory, practice, and teaching of professional communication in critical global contexts such as business, manufacturing, law, health, education, technology, environment, and others. The journal welcomes articles with diverse rhetorical styles, types of inquiry, and contexts of research, but articles are to be submitted in English and grounded in relevant theory and appropriate empirical research methods. The journal is peer reviewed with an editorial board consisting of experienced researchers and practitioners from over 20 countries.

The Journal's main objectives are to:

- develop better theoretical models of global professional communication
- develop a variety of valid and ethical research methodologies for global professional communications
- improve the practice of global business and manufacturing through more effective communication
- improve the professional communication in critical cross-cultural and international contexts

such as the environment, immigration, health, energy, economics, and human rights

- develop inquiries and research agendas that address the most pressing issues and challenges for communicating in the context of globalisation and
- develop better curricula and materials for teaching global professional communication, not only in the United States and Europe, but around the world. Special attention will be given to developing nations.

The journal will be free or *open access*, using PKP open-source software and housed at Eastern Carolina University. The first edition is planned for June 2010, and it will be published quarterly thereafter. Watch for it online at www.rpcg.org. For more information on the journal, contact Barry Thatcher (bathatch@nmsu.edu) or Kirk St. Amant (kirk.stamant@gmail.com).

Barry Thatcher

Founder/Editor-in-Chief, New Mexico State University

Kirk St. Amant

Assistant Editor, East Carolina University

Journal of Technical Writing and Communication

Contents of the current issue

Click the heading below to read the editorial, abstracts and reviews, or to purchase an article or subscribe.

Volume 40, Issue 1, 2010

- From the Editor's Desk, Charles H. Sides, Executive Editor
- Safety Warnings in Tractor Operation Manuals, 1920–1980: Manuals and Warnings Don't Always Work, Elizabeth Tebeaux
- Answering the Call: Toward a History of Proposals, Lisa Meloncon
- Technical Communication Internship Requirements in the Academic Economy: How We Compare among Ourselves and Across Other Applied Fields, Gerald J. Savage and Marcea K. Seible
- The Effects of Integrating On-Going Training for Technical Documentation Teams, Joseph T. Catanio and Teri L. Catanio
- Book reviews

award-winning, authoritative international voice, publishing the latest research by recognized scholars from around the globe

Volume 40 Number 1 — 2010

JOURNAL OF
Technical Writing & Communication

Editor
Charles H. Sides

Baywood Publishing Company, Inc.

Every article ever published in the JTWC is now available in clear, concise, comprehensive PDF format.

New Online-Only Delivery Options

- Enhanced subscription (all articles from volume 1 through current volume)
- Current volume subscription
- Individual articles on pay-per-view basis

Click here for all the details and ordering information.

- Read all article abstracts free
- Download a complimentary sample issue

BAYWOOD PUBLISHING COMPANY, INC.
<http://baywood.com>

Paragraphing: a vanishing art?

Geoffrey Marnell

As traditionally defined, a paragraph is a container for one unit of thought and typically comprises two or more related sentences. Put another way, it expresses one main idea or point and all the sentences in it are related to that main idea or point.¹

In informative writing, it is a good idea to begin each paragraph with a *topic sentence*: a sentence that introduces or summarises what the main idea or point in the paragraph is. This is especially useful to those readers—probably the majority of us—who will skim or scan your text looking just for the main ideas or points.²

Consider this example:

“There are two purposes for paragraphs, the one logical and the other practical. The logical purpose is to signal stages in a narrative or argument. The practical one is to relieve the forbidding gloom of a solid page of text.”³

A new paragraph needs to be indented. Space alone does not always give the reader an unambiguous clue that a new paragraph has begun. Further, space can suggest that a new paragraph has begun when it hasn't.

Note the topic sentence at the start of the paragraph. It clearly tells us what the paragraph is about, and each subsequent sentence is clearly related to it and fulfils its promise.

Another good reason for starting each paragraph with a topic sentence is that it helps crystallise your thoughts. It also reduces the likelihood that you will include in the paragraph other matters that may well deserve a paragraph of their own.

Still, it is legitimate to start a paragraph in other ways. However, with the skimming reader in mind, it is best always to make it clear in the introductory sentence what purpose the paragraph is serving. For example, suppose you start a paragraph with the sentence “An example is illustrated below”. Strictly speaking, this is not a topic sentence, but it does indicate to the reader that the purpose of the paragraph is to illustrate something that was discussed in the previous paragraph. Thus the needs of the skimming reader are met. Another example: “However, there are counter-examples”. Again, this is not, strictly speaking, a topic sentence, but it does indicate to the skimming reader that material discussed in the previous paragraph is about to be contradicted.

Paragraph cohesion and linking words

If a paragraph is a unit of thought expressing one main idea with all the sentences in it related to that main idea, you need to show how the sentences—or more appropriately the ideas in the sentences—are related to one another. When it is clear how all the sentences in a paragraph are related, we say that the paragraph exhibits *cohesion*. The sentences stick together, and reading them provides a fluid, unbroken stream of ideas.

One way of ensuring a fluid, unbroken stream of ideas is to introduce the second and each subsequent sentence with a word or phrase that indicates the link

between it and the sentence before it. Common linking words or phrases are *therefore*, *so*, *hence*, *accordingly*, *it follows*, *then*, *as a result*, *in contrast*, *at the same time*, *however*, *further* and *for the same reason*:

“A number of drugs have shown promise in treating the disease. Doxycycline at 250mg killed 90% of the bacteria. *In contrast*, penicillin at the same dosage killed at best 60%. *However*, penicillin at 500mg killed all the bacteria. *Therefore*, our recommendation ...”

A pronoun, such as *it*, can also usefully link one sentence with another.

“Melanoma is a particularly aggressive form of skin cancer. It kills 1600 Australians every year.”

Indeed, there is no circumscribed set of words or phrases that will work as linking words. The pool you can choose from is vast, so long as your reader can see the development of your argument or description—that is, see how the distinct ideas in each of your sentences are:

- related to the main topic that the paragraph is about and
- related to each other (through elaboration, extension or qualification).

Paragraph styling

In traditional publishing, a paragraph is *indented* (or at least every paragraph other than the first paragraph after a heading is indented). This is the style that has been adopted in this bulletin. However, the widespread contemporary practice is not to indent paragraphs, but to separate them with a blank line (or with extra space above or below the paragraph).

The contemporary practice is flawed, and for two main reasons. First, if a paragraph falls at the bottom

1. *The Penguin working words: an Australian guide to modern English usage*, Penguin, Ringwood, 1993, p. 390.
2. Pam Peters, *The Cambridge guide to Australian English usage*, Cambridge University Press, Cambridge (UK), 2007, p. 594.
3. Nicholas Hudson, *Modern Australian usage*, Oxford University Press, Melbourne, 1993, p. 294.

of a page and its last sentence finishes close to the right margin, there are no obvious clues for the reader to interpret the next sentence—starting at the top of the next page—as the beginning of a new paragraph (if that is what it is). If you have a *space after* or *space before* setting as part of your style definitions, the space is ignored in both Microsoft Word and Adobe FrameMaker. The first line of the next block of texts starts flush against the top margin *regardless of your space settings*. If you use the `ENTER` key to manually add space between paragraphs—a common though unwise practice—you will get a blank line at the top of the next page. But how can you expect the reader of a print-out of the document to interpret that space as space *in addition to the normal space above the main text frame (that is, in addition to the height of the top margin)*? Readers don't keep in mind the exact height of the top margin so that they can easily detect variances, and most of us wouldn't be able to anyway by eye alone. Moreover, you can't expect the reader to look across at the facing page in order to judge whether both top lines are level. That is simply not how people read. Even so, it would only provide useful information if one, *but not both*, top lines represented the start of a new paragraph.

The first-line indent of traditional publishing overcomes this problem.

The first-line indent overcomes another problem inherent in the contemporary practice of using just space to separate paragraphs. This occurs where you have *block text*, that is, text set off from the main body of the paragraph (as in the following example):

This is an example of block text, text set off from the parent paragraph but still part of the parent paragraph.

If space is the only guide a reader has as to when a new paragraph begins, then it will not be obvious to the reader when the text below a piece of block text begins a new paragraph. Note that this particular block of text is not indented, which should make it clear that it is the continuation of the paragraph that began just before the block text—but only because of the convention adopted in this bulletin of indenting new paragraphs (other than the first one after a heading).

You can automate most indenting with appropriate style definitions. Simply specify that the paragraph style to follow each heading style is one without a left indent (say *body_full_out*) and specify that the paragraph style to follow the *body_full_out* style is one with a left indent (say *body_indented*). Finally, specify that the style to follow *body_indented* is *body_indented*.

Paragraph length

A paragraph can have any number of sentences. However, it's best to avoid the one-sentence paragraph, increasingly common in newspapers.¹

It is difficult, if not impossible, to develop a thought or idea in just a single sentence. And if you continue the development of that one thought or idea across a number of paragraphs, you run the risk of introducing ambiguity. This is partly because readers will be expecting subsequent paragraphs to be about another idea or thought. It is also because readers have no immediate cues that, say, sentence six is still talking about the topic introduced in sentence one or is talking about a new topic altogether.

One-sentence paragraphs are also inimical to the skimming reader (which is most of us). Where every paragraph is just one sentence, skim reading—the reading of the first sentence in a paragraph—is tantamount to reading the *entire* document. Faced with a document entirely of one-sentence paragraphs, the busy reader is likely to abandon reading before long.

On the other hand, long paragraphs can become overwhelming. They might also amalgamate more than one thought or idea, thereby denying the skimming reader a true glimpse of the main development or argument.

So what is an appropriate length?

“For general purposes, paragraphs from 3 to 8 sentences long are a suitable size for developing discussion, and some publishers recommend an upper limit of 5/6 sentences.”²

Composing a paragraph

How does one decide what should go into a paragraph and what is best placed in another paragraph? The common definition of a paragraph as a *unit of thought expressing one main idea with all the sentences in it related to that main idea* is a little vague, and thus of limited practical use. We could flesh out the definition to:

A paragraph is one sentence that expresses the main topic to be discussed, together with other sentences that elaborate, expand on or analyse that main topic.

This is a stronger definition and provides some guidance as to what should go into a paragraph, but it still has practical limitations.

Unfortunately there is no widely accepted formula or algorithm you can apply to determine whether a sentence belongs in one paragraph or another. But if we combine these definitions—with their emphasis on a paragraph necessarily being a container for one main idea or thought—with two facts about how people read, we can pull out of the

-
1. The one-sentence command (or step) in a procedure is a notable exception. The practice of separating the command from the result, and from other explanatory notes, generally improves the efficiency with which tasks are done.
 2. Pam Peters, *op. cit.*, p. 595.

mixture an algorithm of some practical use. The extra two facts we need to acknowledge are:

- readers find long paragraphs uninviting, and even daunting; they find it easier to process new information if it is provided in chunks
- readers commonly skim or scan, reading the introductory sentence before deciding whether to continue reading the paragraph or to skip to the next.

Bearing all this in mind, the following approach—illustrated in figure 1 on page 12—might be useful in many forms of declarative or informative writing (although certainly not in novel or story writing). It assumes that you have just written the heading of the section you want to write about and are about to compose the first paragraph after that heading.

The technique

1. Write a topic sentence, that is, a sentence that introduces or summarises what the main idea is in the paragraph you are setting out to write.
2. Is the topic sentence conceptually related to the topic suggested by the heading of the section?

If the topic sentence is not conceptually related, delete it and start again.

This step prevents you lapsing into journalism, where colour, context, characters and setting are

given as much prominence as the facts to be got across. Here is an example:

“Quasars and black holes

“When I was growing up in the Goulburn Valley, there were few clubs or societies for children interested in anything other than sport.”

In technical and scientific writing, facts always take centre-stage. Scene-setting, especially of an autobiographical nature, is not directly relevant to the needs of the typical reader.

3. Write another sentence.
4. Is this sentence strongly relevant to the topic sentence?

If the sentence is not strongly relevant, move it to another paragraph (or delete it) and then repeat from step 3 above. If the sentence is strongly relevant, go to step 5.

5. Are you intending to give the sub-topic introduced with the new sentence in-depth treatment right now (more than two or three sentences)?

If not, leave the sentence in the paragraph for now and go to step 7; otherwise go to step 6.

6. Are you going to give other sub-topics you intend to add to the paragraph similar in-depth treatment?

If so, move this sentence to a new paragraph and continue from step 8. In-depth treatment of two or more topics is best spread across several paragraphs rather than crammed into one long paragraph. Remember: readers shun long paragraphs.

7. Is the sentence crucial to the overall description, argument or analysis being expanded in the document (or the current section of the document)?

If it is crucial, move it to another paragraph and consider making it the topic sentence in that paragraph. (Skimmers only read the first sentence of a paragraph before moving on, and thus if your sentence is essential in whatever description, argument or analysis you are developing, then ideally it should be placed at the start of a paragraph.)

If it is not crucial, leave it where it is.

8. If you have more to write about the main topic of the paragraph (as expressed in the topic sentence) then continue from step 3 above. Otherwise review the paragraph for logical sequence and cohesion and then move on to the next paragraph.

The image shows a blue rectangular advertisement for the Commercial Translation Centre. At the top, the words "commercial translation centre" are written in white, with a large stylized "S" logo to the left. Below this, the text "For Effective Translation/Localisation Of Technical Documentation In 80 Languages" is displayed in white. At the bottom, contact information is provided: "Australia Wide Service Since 1988", "Call 1800 655 224 (Australia Toll Free)", "Email: mail@ctc.com.au", and "Web: www.ctc4.com". The "commercial translation centre" name is repeated at the very bottom in white.

commercial translation centre

For Effective Translation/Localisation Of Technical Documentation In 80 Languages

Australia Wide Service Since 1988
Call 1800 655 224 (Australia Toll Free)
Email: mail@ctc.com.au
Web: www.ctc4.com

commercial translation centre

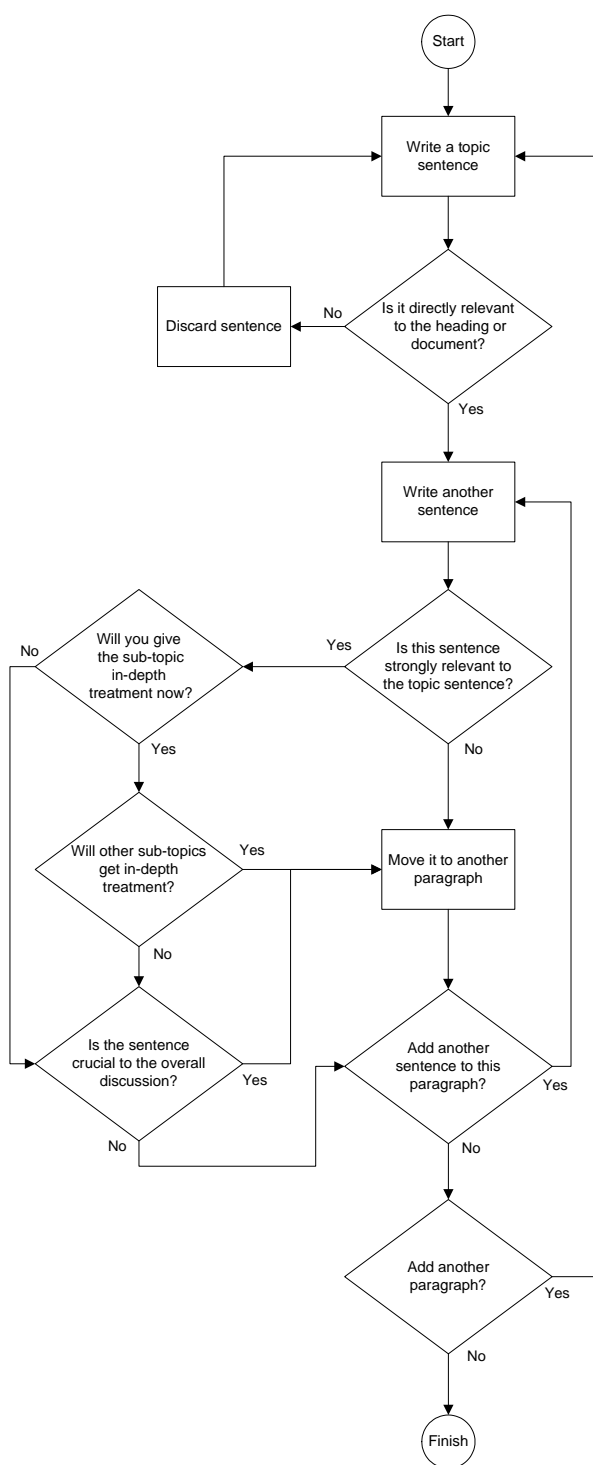


Figure 1: Composing a paragraph

An example

Setting balloons aloft [section heading]

One way to set a balloon aloft is to fill it with hydrogen gas. [Topic sentence: directly relevant to heading? Yes. So leave the sentence where it is.]

There are a number of ways of generating hydrogen gas. [Strongly relevant to topic? Yes. But this sub-topic is not going to get in-depth treatment, and the sentence is not crucial to the overall development of the information. In other words, if skimmers miss this

sentence, there will not be a significant gap in their understanding of the ways of setting a balloon aloft. So leave the sentence in the paragraph.]

One simple way is to use household caustic soda and aluminium scraps. [Strongly relevant to topic? Yes. And this sub-topic is going to get in-depth treatment, but no other way of creating hydrogen is going to receive in-depth treatment in this paragraph (otherwise this sentence would be moved to a new paragraph). Further, the sentence is not crucial to the overall development of the information. So leave the sentence in the paragraph.]

Place three or four tablespoons of caustic soda in a glass bottle with a narrow neck. [Strongly relevant to topic? Yes. And this sub-topic is not going to get in-depth treatment nor is it crucial to overall development of the passage. So leave the sentence in the paragraph.]

Another way to set a balloon aloft is to fill it with helium. [Strongly relevant to topic sentence? No, the topic sentence is about filling a balloon with hydrogen. So this sentence belongs in another paragraph.]

Another approach

Setting balloons aloft [section heading]

You can set a balloon aloft by filling it with a gas that is lighter than air. Two such gases can be readily produced in the amateur chemist's laboratory: hydrogen and helium. [So far so good. The topic sentence is relevant, and the second sentence is directly elaborating on the topic sentence.]

To fill a balloon with hydrogen gas ... [The relevance of this sentence to the topic sentence is rather weak. The topic sentence is about how to get balloons to rise, but now we are talking about how to fill a balloon with hydrogen. So this sentence needs to be moved to another paragraph.]

To fill a balloon with helium gas ... [The relevance of this sentence to the topic sentence is rather weak. The topic sentence is about filling a balloon with hydrogen, but now we are talking about how to fill a balloon with helium. So this sentence needs to be moved to another paragraph.]

...

Good paragraphing: the skimming test

The skimming test is based on the fact that most people skim scientific and technical documents, only reading the paragraphs that are of special interest to them. Skimming is a two-step process: read the first sentence of a paragraph and then decide whether to continue reading or skip to the next paragraph. More often than not most readers will skip at least one paragraph, and usually more. So, if you want to maximise your chance of being fully understood by the maximum number of people who look at your document, make sure that:

- the main point of each paragraph is made in the first sentence (so that the first sentence is the *topic sentence*) and
- all the first sentences, when considered together, provide a satisfactory summary of what you have written.

So the skimming test is this: once you have completed writing a section of your document, read through it *but just read the first sentence of each paragraph*. As you are reading, ask yourself:

Do all the first sentences, when taken together, adequately portray the gist of the section of the document I have just written? In other words, would they, when strung together (and tweaked with linking words), provide an adequate summary of what I have written?

If the answer is no, then you should reconsider your paragraphing.

For example, if you are presenting an argument and it has four premises, it is unwise to present three of the premises as first sentences and embed the other premise within a paragraph (rather than at the start of a paragraph). The skimming reader will not see the full skeleton of your argument and may think that you have not proved your point even if you in fact have.

Likewise, if you have been describing how to send a balloon aloft by filling it with a gas that is lighter than air and you have described *in roughly equal depth* a number of techniques for producing such a gas,

introducing one technique via a first sentence but not the other will not pass the skimming test. For example, the following list of first sentences would not provide an adequate summary if a method of filling a balloon with hydrogen gas was discussed in addition to the method of filling a balloon with helium gas:

You can set a balloon aloft by filling it with a gas that is lighter than air.

To fill a balloon with helium gas ...

You can track the balloon using a simple radio transmitter.

There are a number of ways you can improve your chances of retrieving a spent balloon.

In this case, there needs to be a paragraph with a sentence starting *To fill a balloon with hydrogen gas ...* or the like.

Note that while your set of first sentences should provide an *adequate* summary of what you have written, they may not necessarily provide a *good* summary. There may be any number of paragraphs that are providing merely secondary or illustrative material — analogies, parallels, examples, case studies and so on — paragraphs that could be deleted without thereby breaking the basic logic of the text. A good summary of that material would not include the gist of the secondary or illustrative material.

However, the skimming test is designed only to ensure that all the main points will be picked up by the skimmer, not that the skimmer will only encounter your main points.



First published in *The Age*, 31/10/2009. Reprinted with the kind permission of Michael Leunig. © Michael Leunig 2009.

Paragraphing quotations revisited

In a previous issue of *Words* (volume 1, issue 1), I noted that a number of contemporary language manuals are advising that quotation marks are not needed around a *block quotation*, that is, a quotation set apart and set differently from the text that introduces it. A block quotation is often indented from the left and often in a different font (or a smaller font size) than body text. I went on to note that where there are other types of content that you want to set similarly (such as examples) it will help the reader detect immediately that a block of text is a quote and not something else if you enclose it in quotation marks.

But even if there are no other content types that you want to set in a similar way to block quotations, there are still good reasons for setting block quotations in quotation marks. For example, suppose you start a document with a quotation. If the quotation takes up most or all of the first page of the document, the reader will have no obvious clues right away that what they are reading is a quotation. They might only find this out after turning the page. Thus they start out thinking that what they are reading are the author's views only to discover later

that it was the views of someone the author was quoting. An example can be seen on page one of *Radical Hope* by Noel Pearson.¹

This problem can occur on other pages too, not just the first page. For example, it will occur whenever a block quotation of considerable size starts at the top of any page. Without body text in the immediate vicinity to provide a ready comparison, the eye cannot easily detect the usually small indentation (one or two em spaces, perhaps) and the slightly smaller font size (usually just a point). Thus it is all too easy for the reader to think, wrongly, that what they are reading are the author's views. It is distracting for the reader to subsequently have to disentangle the views of quoted sources from the views of the author. An example can be seen on page five of *Radical Hope* by Noel Pearson.²

It is best, then, to continue using quotation marks for block quotations (just as they are used for direct quotations in running text).

Geoffrey Marnell

1. *Quarterly Essay* 35, Black Inc., 2009.
2. *ibid.*

Journal



quarterly
52 full colour pages
£45 a year (£10 off for ASTC members)
contribute: journal.editor@istc.org.uk
archive (ISTC members only): www.istc.org.uk/Members_Area/communicator_archive.htm
subscribe: istc@istc.org.uk
advertise: felicity@tou-can.co.uk

Broaden your horizons
with subscriptions to

periodicals

Institute
of Scientific
and Technical
Communicators
Newsletter



monthly
PDF (website or e-mail)
freely available at no charge
contribute: newsletter.editor@istc.org.uk
archive: www.istc.org.uk/Publications/Newsletter/newsletter_archive.htm

Looking at DITA as just another tool

Suchitra Govindarajan

I'm not an expert on DITA, but I consider myself well-qualified to write about learning it. Over the past few years, I've made a number of attempts to learn DITA, both on my own, and with some help. A complete grasp of the subject eluded me until I did a course on structured authoring with DITA at Swinburne University.

My eureka moment came early in the course. In the space of about 30 minutes, all of us students were able to author simple topics in an XML editor and see them transformed into CHM and PDF files. I thought to myself, is DITA really this easy?

I kept going back to that question during the course, and, every time, the answer was different. There are aspects of DITA that are impossibly complex: publishing to a well-styled and readable output, for one. There are also parts of DITA that are very simple—even for the most non-technical of us. If you choose an XML editor with a What-You-See-Is-One-Option (WYSIOO) view (explained in the next section), it's really not that different from learning any other authoring tool. And, as with any other authoring tool, you can jump right in without knowing anything more than just the basics.

This article is my rough guide for those who would like to jump right in. (Note that we won't cover publishing or anything to do with the DITA Open Toolkit.)



**MACQUARIE
DICTIONARY
ONLINE**

www.macquariedictionary.com.au

Subscribe to the complete Macquarie Dictionary online, updated annually with new words and definitions.

Also available online is the full Macquarie Thesaurus - that perfect word is just a click away.

Try it out now for FREE!

Macquarie Online is offering free extended trial access. Simply contact Macquarie Online to set up your 3 months free access. Quote code: 3mfTrialAC

Macquarie Online Support
phone: 1800 645 349
email: support@macquarieonline.com.au



Australia's national dictionary

Don't start at the beginning

Nearly every DITA tutorial begins with a definition of DITA and a rundown of XML and document type definitions (DTDs). My view on this is similar to my view on prologues in fiction: they make more sense after you've read the book.

Don't worry about what DITA is. It is indeed an XML architecture and a writing methodology. It might even be a new paradigm. However, these definitions do not really help someone who is attempting to learn DITA. You also don't need to know anything about DTDs or XML schemas, beyond the fact that DITA, by definition, comes with a set of these.

In order to write in DITA, all you really need to know is the following.

- You mark up text using elements and attributes. If you're familiar with HTML, you already know what elements and attributes are. At its simplest, an element is a container for text. An attribute is a descriptor for that container. The rules for elements and attributes in XML are different from those in HTML and are generally stricter.
- You don't get to choose styles. When you author in DITA, you have no control over style information such as fonts and colours. Most XML editors will choose a pretty, formatted look on your behalf. This is called a WYSIOO view. It is not part of your file. It is just a mock-up of how your file could look when published. You'll only see that view while you're in the editor.
- You need to check if your document is valid. Working with DITA means that you're conforming to a standard. Therefore you need to validate every document you author against the standard. Thankfully, most XML editors will either not allow you to create invalid documents, or will alert you when your document is invalid.

What you already know about DITA

A lot of the ideas behind DITA are not new. If you have had a few years experience as a technical writer and have used at least one help-authoring tool, there's a lot you already know. For example:

- **Topic-based authoring** will be familiar to anyone who has used a help-authoring tool such as Adobe RoboHelp or Madcap Flare. You would also be familiar with the idea if you've published a long document in HTML format. DITA takes a more narrow view of what a topic is than most other tools. We'll cover that in the next section.

- **Single-sourcing** has been discussed for many years in the technical writing industry: from one set of files you can create various outputs.
- **DITA maps** are not dissimilar to tables of contents in help-authoring tools or books in Adobe FrameMaker.
- **Content reuse** is familiar to anyone who has used Adobe FrameMaker's text insets or Madcap Flare's snippets.
- **Separation of content and style** is not new to most writers either, even if we have never encountered such a clear and total separation. Many of us have authored raw text in HTML with the styling provided by a separate CSS. Those who have published in different formats will also have experienced authoring in one view and publishing to another.

Know what you want to write about

To get the most out of your learning experience, I would recommend that you select a fairly complex practice project, one that includes everything you would find in a common document—images, cross-references, links and so on.

The biggest difference I found with DITA was that I had to select the kind of topic I was going to write before I started to write it. Changing my mind afterwards was possible, but a bit complicated.

DITA requires you to choose up-front whether you're going to write conceptual material (such as an overview), a task (such as a procedure) or reference information (such as information that would normally be in an appendix). Having made that choice, you then choose a *concept* topic, *task* topic or

reference topic accordingly. Each topic type provides different elements and only those are allowed within it. In other words, you cannot mix two content types in the one topic, such as a concept followed by a task. You need to create separate topics, one for each content type.

Start writing

The two main features you should look for when choosing an XML editor are a WYSIOO view and built-in support for DITA. My recommendations are XMLmind (<http://www.xmlmind.com/xmleditor/>) or XMetal (<http://na.justsystems.com/index.php>).

Support for DITA usually means that you can simply go to the **File** menu, choose **New** and select from a list of DITA topic types (concept, task and reference). You'll also find software aids that list the elements that are allowed at various positions in the document. For example, in XMLmind, you can use the **Insert**, **Insert before** and **Insert after** options to determine what elements are allowed on either side of the current one (see Figure 1 below). You can also access all allowed attributes and values.

In XMLmind, you can also press **SHIFT + F1** at any time to look up the DITA content model. The content model is a bit technical but, once you learn to parse it, you will find useful information about DITA.

Write as you would normally, selecting the right elements and attributes for your needs at every stage. I like switching to other views—such as the structure or code view—to see the effect of additions and changes. Remember to validate your document frequently.

When you have created a few topics, you can also look at creating DITA maps and publishing them.

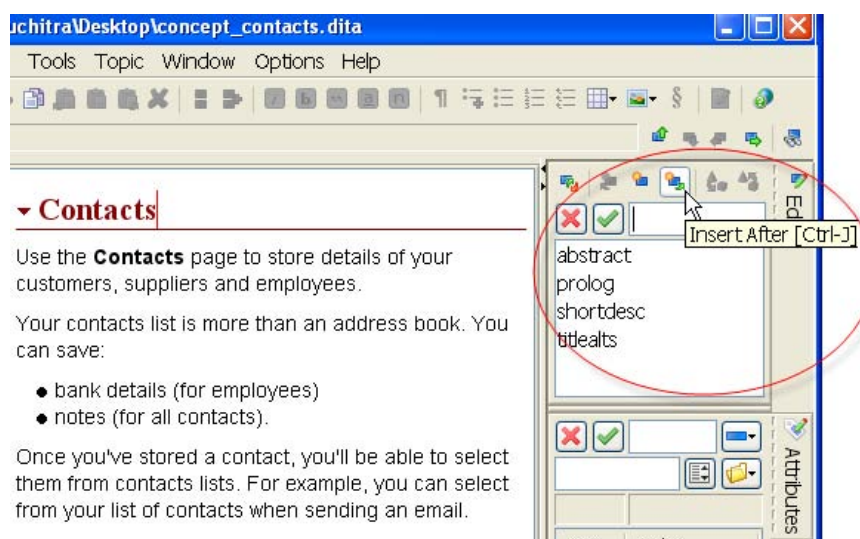


Figure 1: Using XMLmind's DITA aids

Help with writing

If you find writing in DITA difficult, consider looking at sample DITA files. There are some available at http://dita-ot.sourceforge.net/SourceForgeFiles/doc/user_guide.html.

Better still, follow the tutorial that is part of the excellent *DITA for Solo Writers* guide at <http://www.ditausers.org/tutorials/lone-dita/ditaguide.pdf> (especially the “Creating Content” section).

Conclusion

In the early days of the internet, it was a matter of pride to have coded your website solely in Notepad. People who used software like Microsoft FrontPage were looked down upon.

Cut to the present world of instant blogs and websites, where you don’t even need to know that you’re using HTML.

On one hand, I am excited by DITA, by its sheer elegance. On the other hand, when I read some of the DITA documentation, I feel like I’m about to drown in jargon. I look forward to the time when we’re all authoring in and publishing from DITA, but have the ability to choose how deep our knowledge should go. Just as it is now with HTML.

Suchitra Govindarajan

Suchitra Govindarajan has been a technical writer since 2000, and has worked in India and Australia. She hates misplaced modifiers and anything made with eggplant.

Book review

Nikki Ward

Dying Words: Endangered languages and what they have to tell us by Nicholas Evans (John Wiley & Sons, Richmond, 2009, \$49.95)

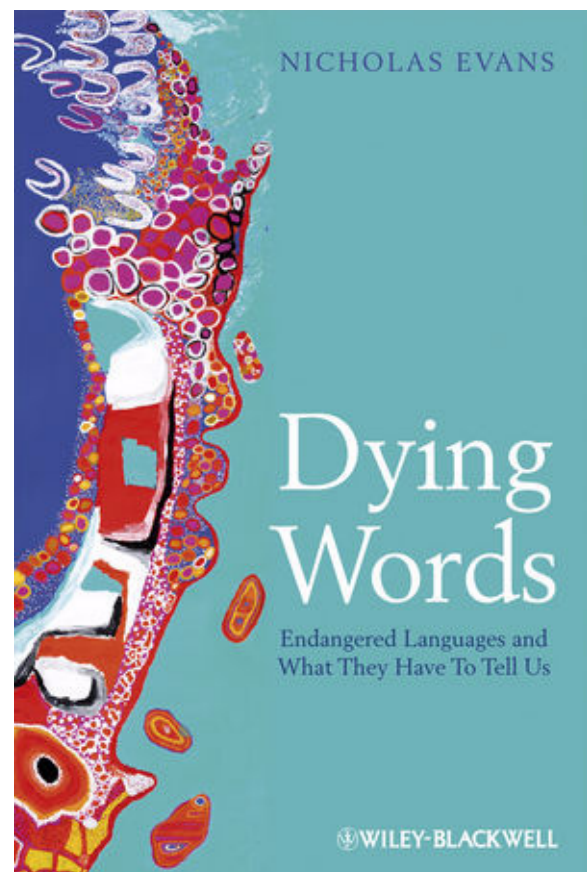
When I first heard about this book many questions surfaced: “Why on earth do we have to be concerned about language death? What is it that different languages brings us in terms of a collective human legacy? Why does it matter? Or does it matter at all?”

Then you read descriptions of this book that go along the lines of:

“The next century will see more than half of the world’s 6,000 languages become extinct, and most of these will disappear without being adequately recorded. Written by one of the leading figures in language documentation, this fascinating book explores what humanity stands to lose as a result.”¹

Being a multilingualist myself, and a speaker of a dying language (Afrikaans), I felt a certain fascination in finding out what Professor Nicholas Evans had to say about the importance of language preservation. My first impression from the book reviews was that this would be an easy read, especially since I was already keen to consume its contents. However, when I received the book I could see that this was, in fact, a very detailed academic work. That said, it is written simply enough for the layman to understand, and I found that I was enchanted by the substance of its content. In places, and I hasten to add only in places, did I find the text became a little too complex (in terms of linguistics theory).

As a whole, the text promotes a strong awareness of language origin, spread, endangerment and death, with a vast array of solid examples to back up the assertions made. The most striking aspect of this work is how Evans demonstrates language at work within culture as an expression of our thoughts and hence our reality. He expands on William von Humboldt’s (1767–1835) theory that one can only develop a “comprehensive recognition of an objective world” through the combination of all languages used to describe the world. No one



1. Reiter’s Scientific, Professional Technical Books, <http://www.reiters.com/>, Viewed 3 January 2010.

language alone can give you this. Evans posits that language exposes new modes of thought, giving us a “new lens” to perceive the world through. He theorises that language and thought act in tandem to create our reality, through a thought process known as *convergence* and another phenomenon known as *linguistic relativity*.¹ Given that our thoughts and hence our language create our reality, Professor Evans states:

“... there may well exist a completely different but internally coherent model of the physical universe, right down to the base ontologies of space and time.” (p. 162)

I found myself deep in contemplation after reading this particular chapter. I realised that each language—with its unique philosophy, knowledge and cultural assumptions—gives us access to new doors of perception (so it seems that Aldous Huxley was onto something, although I hasten to add, my epiphany wasn’t drug-induced!)

Evans goes on to state that languages, as phenomena, have a direct impact on our collected intellectual heritage. Further, he attributes the demise of indigenous

linguistic diversity to European colonisation and to the elites in the nation-states they created. For example, much colonisation was carried out by countries steeped in the Judeo-Christian tradition, a tradition that attributes the profusion of languages to God’s wrath at man’s attempt to build a tower that would reach to Heaven (the Tower of Babel). Such a belief, Evans contends, was the basis for colonists to actively discourage the speaking of indigenous languages. From the indigenous perspective, however, linguistic diversity was akin to a passport allowing its speaker to hold identity and to make claim to country. It vested them with authority and spiritual security and gave them access to hidden knowledge. Sadly this aspect of our collected cultural heritage has been severely impoverished as a result of cultural genocide, but that said, it is evident that people like Evans are doing some good work in this area by documenting languages on the endangered list.

With reference to the hidden knowledge that language carries, Evans shows how words are used in indigenous languages to describe ecological links between plant and animal species that in turn made the transmission of empirical knowledge easier and more accurate. My favourite example is where the discovery of Prostarin, effective against HIV-type 1,

can be traced to a conversation between a Samoan translator (a missionary’s son), a Samoan tribal healer and an ethno-botanist. Without access to the language, that knowledge may have stayed buried for a while longer, and how many lives would have been lost as a result?

It’s not all bad though. Language death does have an upside. Interestingly enough, language death can also give rise to language birth (or *rebirth* to be more accurate). Scripts and texts that have been buried for centuries are rediscovered and then reconstructed to give historians an inside picture on what life was like at that time, and for linguists, an idea on how the language of that time functioned grammatically. Here Evans looks at how written records can experience multiple deaths due to physical destruction, misplacement or decay—or, in some cases, overwriting, thus giving rise to palimpsests. I had to look this word up, but basically it is a medium for writing—such as a parchment—from which the

writing has been partially or completely erased to make room for another text. (In other words, manuscripts were reused several times, mostly due to resource shortages.) In early Christian times, a single folio of

parchment—a sheet folded so as to yield four pages—required the skin of an entire sheep. To create a book similar to the one I am reviewing would have required the slaughter of around 90 sheep!

The process of language reconstruction is not a simple one and at best, unwieldy. Evans outlines a six-step plan that allows linguists to reconstruct undeciphered scripts. However, the resurrected language and culture is impoverished because the reconstruction lacks the richness of its original form, much like the reconstruction of a digital wave form from an analog wave form.

Evans also looks at the touchy subject of biblical text translation. This is where I tried my own hand at looking at the metadata of a direct translation that was used for a number of other biblical translations (for example, the English King James translation, and the Good News for Modern Man translation). I could see where the margin for error and the room for improvement sat. This process is extremely subjective and exposes how different words were supplanted to correlate with what the destination of the translation is for, namely, a particular religion. The translation did not adhere to the actual textual meaning but rather to what people wanted it to say. For example, one translation states:

“... in perils of robbers, in perils by mine own countrymen, in perils by the heathen ...”

To capture languages allows us to preserve modes of thought that we could not otherwise access with just one language.

1. See BL Whorf, *Language, thought and reality*, MIT Press, Cambridge MA, 1956.

But this appears in the Good News for Modern Man as:

“... in danger from fellow Jews and from Gentiles ...”¹

As subtle as this may seem, this blatant word replacement makes all sorts of assumptions about what was originally intended. Draw your own conclusions!

In closing, Evans agrees with Tomasello that:

“... new forms of cultural learning create the possibility of a kind of ratchet effect in which human beings not only pooled their cognitive resources contemporaneously, but they also built on another’s cognitive inventions over time. This new form of cultural evolution thus created artifacts and social practices with a history, so that each new generation of children grew up in something like the accumulated wisdom of their entire social group, past and present.”²

In my opinion this is the crux of the book: what legacy are we offering to humanity? To capture languages allows us to preserve modes of thought that we could not otherwise access with just one language. Without language diversity we rot in stagnant waters and place our species in danger of group-think, where only one mode of thought transports us to the he-with-a-hammer-thinks-everything-is-a-nail territory. I for one, see Afrikaans as a dying language in South Africa. Sadly, I don’t speak it much in Australia, yet it gives me

1. 2 Corinthians 11: 26–7.

2. M Tomasello, *The cultural origins of human cognition*, Cambridge University Press, Cambridge, 1999, p. 527.

tremendous double-vision when it comes to placing cultures side-by-side. This is where the juice is: it allows for creative thinking, and, as Einstein once said, “the secret to creativity is knowing how to hide your sources”.

Double-vision aside, why would I want to preserve Afrikaans? Well, because there is an inherent beauty and comedy in Afrikaans expression that I just love. Sadly, this gets lost in translation into English, so there is no point in trying to explain the following example. But for those who understand Afrikaans, I read recently about two game rangers who were chased by a lioness while riding motor-bikes around the Kruger National Park. One ranger describes it as follows: “... nadat die lieue vyfie moeg raak om agter ons te hol, het ons motorfietse eendelik tot rus gekom. Toe ons van ons motorfietse afklim het ons kniee skoon inmekaar geswak.” Basically, their knees wouldn’t function (they sagged) when they got off their bikes as a result of being chased by the lion, but it is so much funnier in Afrikaans.

Dying Words has been a tremendously worthwhile read and it took me on quite a journey, both geographically and mentally. For anyone interested in linguistics or simply fascinated by how languages work—from borrowed words to language evolution to the preservation of language—the value of the content of this book far outweighs its cover price.

Nikki Ward

Nikki Ward works as an IT documentation and training analyst creating training programmes, e-learning resources and user guides. She also has an associate degree in writing and is passionate about languages of any description, from Auslan (sign language) to Hebrew. She can be contacted at ezistep@gmail.com.

Don’t miss this great professional development and networking opportunity!

**12th to 14th May 2010
Darwin, NT, Australia**



20 Sessions... 10 Expert Speakers... Pre-Conference Workshops... Social Events... Networking...

Knowledge Management, DITA, eLearning, Help, Structured Documents, Wikis, Case Studies, more...

Keep your skills up-to-date, and **learn new techniques, tips and technologies**. If you missed a previous AODC, don’t make the same mistake in 2010!

The biggest annual event for technical communicators from Australia and New Zealand

Register now at www.aodc.com.au

Proudly presented by



Miscellany

Poetry and technical writing: an odd couple?

Back in the 1980s and early 1990s, a number of academic papers were published linking poetry and technical writing. No laughing: it's true. There were claims that both disciplines used similar language structures and communicative techniques, and that academic studies in poetry provided a good basis for teaching technical writing.¹ I'm sure that there are some contemporary technical writers who would be disappointed to learn that that thread of academic enquiry died out fairly quickly. *An Ode to Microsoft Word* just wouldn't cut it with today's economic rationalists, with their graceless utilitarian imperatives. And perhaps not with readers, too.

But that doesn't mean, of course, that one can't wax lyrical about technical writing itself, even if it would, on reflection, be rather silly to wax lyrical about a steam iron or bookkeeping software. Indeed, there are many technical writers passionate about poetry, and some have even had poems published. That obviously puts paid to the belief that a career that calls for simple, plain, utilitarian English day in day out invariably blunts one's vocabulary and saps one's creative juices.

So here's a call to all poets and would-be poets, those who sublimate their verbal creativity in the quotidian mundaneness of technical writing. The next issue of *Words* will have a poetry section, and the editorial team invites you to submit a poem related, in some degree or other, to technical writing. The deadline is 30 March, and only one potential contributor is blacklisted: *Anonymous*. Send your contributions to words@abelard.com.au.

But what is a sentence?

In 1975, Timothy Fullerton published *Triviata: A Compendium of Useless Information* in which he claimed that an 823-word sentence in Victor Hugo's *Les Misérables* is the longest sentence ever published. Since then, numerous longer sentences have been uncovered. For example, there is a sentence of 4 491 words in James Joyce's *Ulysses*. And a 2001 novel by Jonathan Coe—*The rotters club*—sports a 13 955-word sentence. Impenetrable stuff, indeed.

But hang on a tick. How have the sentences in the works of these long-winded word-sadists been identified? By the appearance of a concluding full stop, it seems.

But if a sentence is simply a string of words marked out with special punctuation—such as a full stop²—then what of those languages that don't use any punctuation (such as Sanskrit and Ancient

Greek)? Are there no sentences in Sanskrit, or is every self-contained text one entire sentence?

On the other hand, if a sentence is defined, somewhat more intuitively, as the container for a discrete chunk of independent information³, then much of what we commonly write and think of as a sentence is not a sentence at all. Consider, for example, *The pound is rising and the dollar is falling*. This has all the appearances of a single sentence, but it contains two chunks of information, each in a separate clause and each capable of being expressed independently.

So, as for the verbal runniness of Hugo, Joyce and Coe, they either wrote strings of words that were not sentences—concatenating clauses into something part-way between a sentence and a paragraph—or they need to give up their titles as Kings of Verbosity to the writers of Sanskrit and Ancient Greek. Or maybe our definitions of *sentence* need tightening.

Emetic writing

"Reporting to the director, the role facilitates the work of the HR functions, continuously improves strategies and procedures, while interfacing the internal and external environments."

Yuk. This is from an advertisement for a "Manager – Administration" at Monash University, one of many examples of gut-turning managerial speak quoted in Don Watson's new book *Bendable Learnings*⁴. Watson is an indefatigable critic of the puffery and obfuscation of the language of governments, bureaucrats and managers. His new book is a collection of many hundreds of examples of such language. Here is another:

"Ford in Australia will right-size our business by staff-balancing changes ..."⁵

Yep, we're going to sack some staff, but, boy, we're not going to make it that damn obvious.

Watson's collection is both side-splitting and frightening, and no one should be distracted from the message by the sloppy editing of Watson's own words, editing that missed a clanger like this:

"It is the principle task of the manager ... to limit the expression and consequences of this ..."⁶

Vale

Vale [Naomi Hall](#): 1972–2009. Technical writer and special friend to many.

1. See, for example, John S. Harris, "Poetry and technical writing" in *Journal of technical writing and communication*, vol. 23, no. 4, 1993, pp. 313–31.

2. See *Style manual for authors, editors and printers*, John Wiley, 2002, p. 97.
3. www.macquariedictionary.com.au (Macquarie Dictionary Online). Viewed 18 September 2009.
4. D Watson, *Bendable learnings: The wisdom of modern management*, Knoff, North Sydney, 2009, p. 87f.
5. *ibid.*, p. 91f.
6. *ibid.*, p. 10.

Mindstretchers

Geoffrey Marnell

Devilish sequences

What number comes next in the following sequence?

1 4 9 16 ?

And what about this one?

3 -112 69 1003 -47 ?

These are strange times indeed!

A woman born in 1896 has had just five birthdays by 1 April 1920. How is that so?

Solutions will appear in the next issue of *Words*.

Solutions to the last puzzles

Puzzle 1

Consistent with the clues given in the puzzle is the possibility that Craig beat Allan 4 times. This seems, at first, paradoxical, since Craig would then have beaten Allan more times than Allan would have beaten Craig despite the fact that Allan beat Bertram more times than Bertram beat Allan, and Bertram beat Craig more times than Craig beat Bertram. But here's proof that it is possible.

Letting A stand for Allan, B for Bertram and C for Craig, the following table represents the 6 possible ways the three could be placed in a race:

	1st	2nd	3rd
a	A	B	C
b	A	C	B
c	B	A	C
d	B	C	A
e	C	A	B
f	C	B	A

We are told that possibility **a** occurred 3 times, and since it was the most consistent result, no other result occurred 3 or more times. This leaves us with 35 combinations of these possibilities representing exactly 7 results (e.g. **3a, b, c, d** and **e; 3a, 2b, c** and **d**; and so on). Rather than working through each of these combinations and noting, for each, the number of instances where Craig has come home before Allan—a long and tedious process—you can employ a little logic to make life easier. You will note that each of possibilities **a, b** and **c** has Allan beating Craig, while each of possibilities **d, e** and **f** has Craig beating Allan. We need only consider, then, combinations of these possibilities where members of the latter set (**d, e** and **f**) appear more frequently than members of the former set. We cannot, of course, omit **a** (which occurred in 3 of the 7 results), but we

can omit both **b** and **c**. Consider, then, the following exhaustive combination of results in which neither **b** nor **c** occur:

$$3a + 2d + e + f$$

$$3a + d + 2e + f$$

$$3a + d + e + 2f$$

In all cases, Allan would have beaten Bertram more times than Bertram would have beaten Allan, and Bertram would have beaten Craig more times than Craig would have beaten Bertram. So the major condition set in the puzzle is met. But in addition, each combination gives Craig 4 victories over Allan which, given the fact that Allan beat Craig at least 3 times out of 7 races, is the maximum possible number of times Craig could have beaten Allan. QED.

Puzzle 2

You may have guessed, given the vast number of possible arrangements of the slips of paper, that there must be a short cut to answering this puzzle. And there is. A little experimentation will show that a number is divisible by 3 if the sum of its individual digits is divisible by 3. Consequently, the order in which the slips of paper are drawn out of the bowl is unimportant. For the sum of the digits in the numbers 1 to 20 (which, incidentally, is not the same thing as the sum of the numbers from 1 to 20) is 102. And since 102 is divisible by 3, any number composed—in any order whatsoever—of the 31 digits that make up the numbers from 1 to 20 must itself be divisible by 3. So the probability that the 31-digit number formed is divisible by 3 is 1: a dead certainty. QED.

AUSTRALIAN STYLE

A NATIONAL BULLETIN ON ISSUES IN AUSTRALIAN STYLE AND ENGLISH IN AUSTRALIA

Announcing the online rebirth of the biannual newsletter published at Macquarie University, featuring:

- articles on language usage and research
- the Feedback questionnaire
- the exclusive crossword Rubicon by David Astle
- cartoons by Judy Dunn
- book reviews, a new word column, and much more

Website: http://www.ling.mq.edu.au/news/australian_style.htm

Contact: adam.smith@mq.edu.au