

TCC Documentation Guidelines



John W. Shipman

2008-01-08 15:08

Abstract

Describes the recommended practices for external and internal documents related to the New Mexico Tech Computer Center.

This publication is available in Web form¹ and also as a PDF document².

Table of Contents

1. Scope	1
2. Where things live	2
3. Software tools for document creation	2
3.1. DocBook	2
3.2. The <i>PyStyler</i> web content system	4
4. Style rationale	5
4.1. General style rules	5
4.2. Web presentation	6
4.3. Print presentation	8
5. The internal help system	9
5.1. When <i>not</i> to use MoinMoin	9
5.2. Limitations and problems of MoinMoin	10

1. Scope

These are guidelines for:

- User-level and internal documentation for Tech Computer Center users and employees
- Online and printed documentation

Note

Before we talk about the best tools for documentation, please keep in mind that *any documentation is better than none*. The TCC documentation staff will be happy to accept Word files, plain ASCII text, crayon on butcher paper, whatever. The important thing is to start with good information, in whatever form, and let the documentation staff worry about editing it and making it attractive.

¹ <http://www.nmt.edu/tcc/doc/plan/>

² <http://www.nmt.edu/tcc/doc/plan/plan.pdf>

2. Where things live

All TCC documentation lives in the TCC homepage structure³, whose pathname is `/u/www/docs/tcc`. This pathname is a soft link to `/fs/teidium/web`, a separate partition that can be mounted read/write from any TCC host.

Documentation resides in three major divisions:

- The TCC homepage and related pages⁴.
- The TCC help system⁵, which is much larger than the TCC homepage. Eventually these two structures will be merged. The Help System came first; the TCC homepage was added later for quick access to the commonest information.
- The TCC Internal Help System is a combination of DocBook documents under directory `/u/www/docs/tcc/ihs`, and online documents in the TCC MoinMoin Wiki (see Section 5, “The internal help system” (p. 9)).

The `/u/www/docs/tcc/` directory, and all directories under it, have their `setgid` permission set, so that all new files and directories are created in group `tcc`. If you work at the TCC, your work account will be in group `tcc`, so you'll be able to create and modify files in this structure.

There are a few core documents, related to internals of the Applications Specialist duties, that live in `/u/www/docs/tcc/doc/`. Eventually these documents will be migrated into the internal help system. Particularly important is the `/u/www/docs/tcc/doc/docbook43/` structure, which contains the DocBook stylesheets and their local customization.

All documents integrated into the main structure will be under full RCS control (Revision Control System) to allow tracking of change histories, control over simultaneous access, and retrieval of earlier versions. The plan is to move to SVN (SubVersion) repositories eventually.

There is a “staging area” for material that has been written and not yet integrated into the regular structures, and there will be a check-in process that maintains a list of all the documents yet to be integrated. The staging area is at `/u/www/docs/tcc/doc/incoming/`.

3. Software tools for document creation

3.1. DocBook

There are times for Web-based documentation, and there are times for physical paper documents. Good Web-based information is the way most people operate nowadays. On the other hand, if the power is off in the server room, you probably don't want to rely on Web-based documentation for the procedure on bringing up servers.

3.1.1. Why DocBook?

The DocBook system gives you the best of both worlds and has these advantages:

- The document exists in a single form that is automatically translated into both Web pages and a paper document.

³ <http://www.nmt.edu/tcc/>

⁴ <http://www.nmt.edu/tcc/>

⁵ <http://www.nmt.edu/tcc/help/>

- To write a document in DocBook, the author doesn't have to worry about exactly how it is presented, just the structure of the document (paragraphs, sections, bullet lists, and so forth).

To learn DocBook, see these TCC documents:

- *Writing documentation with DocBook-XML 4.2*⁶: This explains what a DocBook-formatted document looks like.
- *XML document authoring with emacs nxml - mode*⁷: A good tool for automatically creating valid DocBook documents (or any other XML document type).

Another advantage of DocBook is that all the documents created with it will bear the official TCC logo, and use consistent fonts and layout conventions, so that documents written by different authors will still look like they came from the same organization.

One of the chief duties of the documentation staff is to customize the local presentation of DocBook documents, both in Web and print form, to make them more useable and attractive.

Some people say, why do we need a separate print form? Why can't we just use the print function in our browser to print the web pages? That may work for very short documents, but longer documents need a lot of *navigational features* to help the reader find their way around:

- The *table of contents* of a document shows the overall structure of the document so the reader can decide what parts are relevant, and jump straight to those parts.
- *Cross-references* are vital in larger works. For example, a document describing how to build a graphical user interface should have a section on color, so that all the other sections that discuss how to specify colors can refer to it.
- An *index* is a good idea for book-length documents.

All these features need a way for part of a document to point at some other part. In a Web document, we can use links. But a proper print document should have page numbers, and the table of contents, cross-references and index should use those page numbers. Printing from a browser won't do that.

Using DocBook, the author specifies the cross-references directly in the document, and doesn't even worry about the table of contents or index. When the document is written, it can be translated mechanically into both forms, with links holding the Web form together and the print form structured with page numbers. In either case, the table of contents and index are generated automatically.

3.1.2. Local customization of DocBook

The customization of both Web page and PDF files generated from DocBook documents is controlled by a local customization layer built on top of Norman Walsh's Modular Style Sheets. This customization layer is currently located in directory `/u/www/docs/tcc/doc/docbook42xep`. The root file is `tcc_html.xsl` for HTML, `tcc_fo.xsl` for the print route. Many style features of DocBook web pages use CSS, referring to file `"tcc/docbook.css"`.

The XSLT code for the customization layer is available in a literate exposition: see *Customization of the DocBook XSL stylesheets for the TCC*⁸.

We currently have two DocBook systems in place. The one mentioned above is the production version. It uses a proprietary program named XEP to create the final PDF version of DocBook documents. We don't currently have to pay for XEP, having gotten a free academic license. However, if they ever get

⁶ <http://www.nmt.edu/tcc/help/pubs/docbook42/>

⁷ <http://www.nmt.edu/tcc/help/pubs/nxml/>

⁸ <http://www.nmt.edu/tcc/doc/docbook42xep/ims/>

too expensive, we may have to fall back to the inferior open-source equivalent FOP. Documentation for this system lives in `tcc/doc/docbook42/`.

These documents are essential to anyone who wants to understand or modify our customization layer:

- XSLT, for Extended Stylesheet Language Transforms, is the language used in the stylesheets. A good reference is *XSLT* by Doug Tidwell (O'Reilly, ISBN 0596000537). The official TCC publication *XSLT Reference*⁹ will help you remember XSLT's features once you've learned the language.
- DocBook XSL Stylesheet Reference Documentation¹⁰ by Norman Walsh. This is a fairly skeletal introduction to obtaining, installing and customizing the style sheets.
- *DocBook XSL: The Complete Guide* by Bob Stayton. This is now available as a published book from Sagehill Enterprises, ISBN 0974152110. The TCC purchased a license for the PDF version of this book, and there is a ring-bound hardcopy. This book is absolutely essential for anyone doing serious customization of DocBook.

3.2. The *PyStyler* web content system

We distinguish between *general webs* like the TCC web¹¹ and the TCC help system¹², and *official TCC publications* which live in the TCC publications directory¹³.

- A general web is an informal structure designed to help people find information through browsing. Its primary function is to help people chase links until they get what they need.
- An official publication is a single, unified document on a single subject. Publications are a better medium when the reader needs to be walked through a sequence of related, interdependent topics, such as in a *tutorial*, which is intended for people who don't know how to use some system.

Publications are also better for some kinds of *reference documents*, intended for people who have been through some tutorial or class and just need references to remind them of terms, structures, and such.

Printed versions of publications are handy for situations where the user is away from their computer, or the system is down, or for people who would just rather read from paper instead of a screen.

One way to view a large mixed structure like the TCC Help System is as a pyramid. The user starts at the Help System's homepage¹⁴ and then follows links there to major subject areas, working from large topic areas to more specific topics until they reach the desired information.

However, in practice the upper layers of this pyramid are loosely structured Web pages, with tutorial and reference documents embedded in it.

Currently, a homemade system called *PyStyler* is used to manage the loose Web pages of both the TCC web and the TCC Help System. Documentation:

- *Building informational webs with PyStyler*¹⁵ (PDF only) is the reference manual.
- *Summary of PyStyler*¹⁶ (PDF only) is a quick-reference guide.

This system is rather dated. It expects page creators to use "tag soup", that is, HTML (not XHTML). Until such time as something better comes along, though, *PyStyler* has these virtues:

⁹ <http://www.nmt.edu/tcc/help/pubs/xslt>

¹⁰ <http://docbook.sourceforge.net/release/xsl/current/doc/reference.html>

¹¹ <http://www.nmt.edu/tcc/>

¹² <http://www.nmt.edu/tcc/help/>

¹³ <http://www.nmt.edu/tcc/help/pubs/>

¹⁴ <http://www.nmt.edu/tcc/help/>

¹⁵ <http://www.nmt.edu/tcc/help/pubs/pystyler.pdf>

¹⁶ <http://www.nmt.edu/tcc/help/pubs/pystyler.pdf>

- It enforces a uniform page style, minimizing the amount of work authors have to do to provide navigational links, the TCC logo, and so forth.
- It ensures that all internal links within a given structure are valid. That is, if *PyStyler* runs with no error messages, we can be sure that no links from one Help System page to another Help System page are broken.

The templates that drive *PyStyler's* Web presentation are quite similar to the TCC's Web customization of DocBook. The next section discusses why the author chose that form of presentation.

4. Style rationale

The style of presentation of a document is an art, the art of graphic design.

Different designers may make different choices, but the ultimate test is usefulness. Can the reader find the information? Does it make sense? Is it accurate?

Accuracy is not a style issue. Clarity is not primarily a style issue, but poor style can ruin clarity. The sections that follow discuss the author's decisions and prejudices that went into the current DocBook customization.

4.1. General style rules

A good first book on graphic layout is *The non-designer's design book* by Robin Williams, Peachpit Press, ISBN 0321193857. Many of the principles below come straight out of this book.

4.1.1. Using fonts

A font change should be obvious. In many text fonts, the distinction between normal weight and boldface is relatively subtle. Italics stand out a lot more. When available, extra-bold weights will also work. The idea is to give the reader a big cue and not a subtle one.

The current preferred regular body face is Palatino. It may not be the trendy face, but it's clean and easy to read. The actual face is Palladio, a free public-domain face very close to Palatino.

Headings are currently set in Helvetica for lack of anything better.

For monospaced font, the author is severely prejudiced against the universally used Courier face, considering it one of the ugliest fonts in the history of fonts. Fortunately, the free public-domain Vera Sans Mono is both readable and attractive.

The author's favorite monospaced font is Lucida Typewriter, but this is a proprietary font. The open-source, free Luxi family includes a font called LuxiMono, which is pretty close, but it has one fatal flaw: one cannot tell 0 from O. Vera Sans Mono makes this distinction clear by placing a small hyphen inside the zero; this is not to everyone's taste, but clarity trumps taste here, in the absence of a choice that has both.

The Gnu project has a promising new font named Liberation Mono. It is attractive, but it too suffers from the lack of distinction between 0 and O. The author has received a personal communication around late 2007 from that project acknowledging this problem and promising that it will be remedied in a future version.

Certain elements, such as DocBook's `<application>` element for the names of programs, and its `<guibutton>` element for buttons in graphical user interfaces, are set in a sans-serif face to make them stand out.

4.1.2. Spatial aspects of layout

One of Williams's best general rules is to use empty space to help the reader make associations. For example, in a list of terms with definitions like this:

tree ears

A dried bracket fungus.

fagara

Red Szechwan peppercorns.

Notice how the definition is close to the term, while there is extra space before each term, so the reader can see which two elements go together.

The indentation structure sets the definition off from the term.

4.2. Web presentation

There are currently two places that define the style of the TCC's documentation pages. Relative pathnames starting with `tcc/` refer to absolute path `"/u/www/docs/tcc"`.

- Pages under the control of a *PyStyler* structure use a file called `Template` in the root directory of that structure to define the layout of the web page. Much of the style is specified using CSS in file `"tcc/style.css"`
- The style of Web pages generated from DocBook is specified by the TCC's local customization layer. See Section 3.1.2, "Local customization of DocBook" (p. 3).

The formats of Web pages from these two different sources are very similar. They come from the author's ideas on Web style and navigation, learned from Dr. Jon Price, formerly a professor in the Tech TC program.

Chunking is one of the main principles. Assuming that people can get Web pages served to them without too much delay (and it helps if they are small and light on graphics), it's kinder to the reader to present the information in small chunks. The extreme of this tendency is one chunk per paragraph, but that's probably too far. One older rule of thumb is that the principle content of a page should all be visible without scrolling. There are exceptions, but this is a reasonable first approximation.

Here is one way to look at the parts of a Web page:

1. The *head* contains the page title, TCC logo, and top-of-page navigational links. See Section 4.2.1, "The web page head" (p. 6).
2. The *body* of the page presents its unique content. See Section 4.2.2, "The web page body" (p. 7).
3. The *page-end links* give the reader places to go after reading the content. See Section 4.2.3, "Page end links" (p. 8).
4. The *colophon* comes last. It tells where the page came from, who to contact about it, and where it lives. See Section 4.2.4, "Web page colophon" (p. 8).

4.2.1. The web page head

Right at the very top of the page head is a single line of navigational links. We use text for the links, rather than icons, for two reasons: icons are graphic and slow down page downloads; and icons are not as universally understandable as words, in the author's opinion.

The order of the links is from most important to least important, on the assumption that the reader is from Western cultures that read left-to-right. Here are the links used in both *PyStyler* and DocBook Web pages, with notes on where they differ:

- *Next* links to the next page in sequence, when there is an obvious sequence. On the last page of a sequence, when it isn't clear where most readers will want to go, this link is grayed out.
- *Previous* links to the previous page in sequence. This isn't the same as the browser's *Back* button, because the reader may not have come here from the page previous to it in sequence.
- The next link is *Index* for *PyStyler* pages, and points to that system's automatically generated index of page titles using a KWIC (Key Word In Context) approach. For DocBook pages, this link is called *Contents*, and points back to the table of contents for the document.

The idea in either case is to give the reader a place to jump to see the overall structure of which this page is a part.

- The next link is to the *TCC Help System*, in the hope that someone who is really lost will find that system useful.
- The next two links appear only on *PyStyler* pages: a link called *Publications* that takes them to the TCC publications index, and a *Site map* link that takes them to the *PyStyler*-generated `index.cgi` application that allows them to navigate through the outline of the whole structure.
- Finally, there is a link to the *NM Tech homepage*, so the reader will know what organization hosts this whole structure.

The title tells the reader what this page is about; that's just good basic technical writing.

The TCC logo tells the reader what organization produced this page, and when the reader follows a link to a page without this logo, they know they've reached a page that didn't come from the TCC, and may be less inclined to blame us for any of that page's shortcomings.

These two elements occupy important real estate: the top of the page is where readers look first. Sizing these elements is a balancing act. We want the title and logo to be big enough to read easily, but the more space we take up here, the less is available for content (assuming that we don't want the reader to have to scroll if possible).

4.2.2. The web page body

Set off from the head by a horizontal rule, the main body presents the actual content of the page.

In order to keep page sizes down and their purposes clear, consider these page rôles:

- The purpose of a *routing page* is to give the reader multiple places to go, along with the information they need to figure out which one applies to them. A common organization of such a page is as a bullet list in which each bullet is a choice. Each choice might consist of just a page title, assuming the page title is all the information they need to choose. Or you could have a bullet like "Sky-diving clubs: you must be affiliated with a club to practice sky-diving". The first three words link to a page about sky-diving clubs, and the remaining words nudge the reader, telling them why they might need a club affiliation.

In most multi-page Web structures, the starting page is a routing chunk. Its links may lead to more routing chunks as the reader works from general information to the specific information they need.

- A *leaf page* is the end of a path through routing chunks. It presents some bit of useful information, like explaining a concept, defining a term, or describing an occurrence.
- A *procedure chunk* describes a sequence of steps to be followed.

4.2.3. Page end links

It is rather frustrating to reach the end of a Web page and find no links that take you back off the page. To remedy this, we provide a stereotyped sequence of links that should cover most readers in most situations.

The page-top links are kept on a single line because we want to minimize our use of valuable real estate there. However, once past the page body, space is less critical, so our page-end links give not only a word like “Next” but also the title of the page to which the link is connected. This gives the reader even more information to help them decide whether to follow that link.

Here are the page-end links used in *PyStyler* and DocBook pages. Most of them work the same as the corresponding page-top links.

- *Next*. If there is no obvious next page, this link will not appear.
- *See also*. Dr. Price dislikes the “Up” links that many page authors use, because they assume that the reader has some idea of the overall structure of the document. He prefers that there be an up link, that takes the reader to the next larger containing section, but he feels it should be *called* a “See also” link. Assuming that pages have suitably descriptive titles, the reader will be able to tell whether they want to follow that link.

In some cases there may be multiple see-also links. For example, suppose you are writing a document on car engines. At the end of the section on fuel systems, the author can't really predict what the reader wants to read about next—the exhaust system, the electrical system, the cooling system—so provides a set of see-also links for the different choices.

- *Previous*. If there is no previous page, this link will not appear.
- *Site map* (for *PyStyler* pages only).
- *Index* (for *PyStyler* pages only).
- *Help*: Goes to the TCC Help System.
- *NM Tech homepage*

4.2.4. Web page colophon

Set off by another horizontal rule, the colophon includes:

- The name of the page author or authors.
- The e-mail address where comments or requests should be sent. For most pages this is <tcc-doc@nmt.edu>.
- The last modification timestamp. This is generated automatically by both *PyStyler* and DocBook.
- The URL of the page. This is helpful for browsers whose print function does not show the URL, or for long URLs that don't fit in the heading of printed pages.

4.3. Print presentation

The standard file type for printed publications is PDF (Portable Document Format). Free readers are available for this format.

Here are some notes on the style choices for print presentation.

As with Web pages, the titles should be big enough to stand out, but should not take ridiculous amounts of space.

The TCC logo appears on the first page of every document, and the phrase *New Mexico Tech Computer Center* appears on every page footer, along with the document title and page number.

The author currently prefers to use no page header, leaving more room for content on the page. However, this makes things somewhat crowded in the footer. Also, for some documents (e.g., *Tkinter reference: a GUI for Python*), it may be helpful to the reader to include chapter titles as well as the document title on every page. If this causes problems, the author's first choice for an element to be moved to the page header is the document title and chapter title. This will leave "NMTCC" and the page number in the footer.

The author prefers that section and subsection titles project into the left-hand margin, to make them easier to find.

The author's decision to turn on section numbering (e.g., 5.1.3 for the third subsection of the first subsection of the fifth major section) has been somewhat controversial. The author's feeling is that these numbers help readers keep track of where they are relative to the document structure. If you are in section 5.1.3, you know that section 5.1 is the next higher-level section. These numbers also help readers find a section, providing an alternative to page numbers. If you are looking for section 8, and the page you are looking at has section number 5.1.3, you know you want to go further to find your section.

Verbatim displays, such as program listings or screen shots, are printed over a light gray screen. This gives the reader a clue that the content of these areas is not ordinary narrative text.

5. The internal help system

The requirements for an internal help system are very different than those for external documents.

- Security is essential. Internal documentation must be password-protected so only current TCC employees can even see it.
- Style is somewhat less important. Unlike external documents, we don't care so much about a consistent look and style. However, style *always* matters when it comes to clear presentation of the material.
- Because some people find it tedious to write documentation, we need to make the task as easy as we can.

The current recommendation for the bulk of documentation is our MoinMoin Wiki¹⁷. This system is entirely Web-based, easy to learn and use, and fairly full-featured. To learn this system, see *Using TCC's MoinMoin Wiki for internal documentation*¹⁸. This document covers not only functional aspects of MoinMoin, but local style standards.

5.1. When *not* to use MoinMoin

There is no one perfect system for all needs. Online documentation is handy and easy to create, but some documentations really must be in DocBook for one or more of these reasons:

- Critical documentation, such as power-down and power-up procedures, and disaster recovery notes, must be available in hard-copy form. Up-to-date copies must be maintained in the ring binder on the table in the back right corner of the server room.

It does you no good if you lose the Wiki server and the instructions for putting up the Wiki server are located only there! Even more critically, information on server configuration must be available when the servers are down.

¹⁷ <https://fedora.nmt.edu/tccwiki/>

¹⁸ <http://www.nmt.edu/tcc/ihs/moinmoin/>

- If a document is critical, but is also *sensitive*, it should be in DocBook so that hard-copy is available. However, normally DocBook produces Web pages, but we obviously don't want them up on the Web! In this case, use DocBook, but apply group `tcc` protection to its directory, so that the only form available is the PDF, and that is available only to TCC staffers.
- As nice as MoinMoin is for quickly creating decent documentation, it gets unwieldy for many documents. See Section 5.2, "Limitations and problems of MoinMoin" (p. 10). If you are doing a sizeable document and these things start to bother you, you will find that DocBook has none of these problems.

5.2. Limitations and problems of MoinMoin

Here is a partial list of things MoinMoin can't do, or does badly.

- An important theoretical underpinning of DocBook is to use *structural markup* and not *descriptive markup*. For example, there are several reasons to italicize English text. DocBook has a tag for each reason: `emphasis` for emphasized text; `citetitle` for titles of books, articles, or movies; and `foreignphrase` for phrases in a foreign language.

However, the Wiki italicization markup `' . . . '` doesn't define why the text is italicized. If exported to DocBook, all such text is wrapped in an `emphasis` element, even in cases where that is not the reason for italics.

- If you have a bulleted or numbered list with items that run for more than one line, and any of the lines contain markup such as `' . . . '` for italics, the lines are not wrapped properly. Here is an example that fails:

```

-----
1. If you use 'italic' or 'bold' markup in the first
   line, you can continue the paragraph with explicit line breaks,
   so long as no continuation line contains those kinds of markup.

1. However, if you use any continuation lines that contain
   any 'italic' or 'bold' markup, there will be a line break
   in the output at the same location as the line breaks in the input.
-----

```

As rendered by MoinMoin, step 2 has a line break after the word "contain", so the first line is much too short.

The workaround is to make each bullet or step one huge long line. The Wiki apparently doesn't expect you ever to use the *Enter* key to break up long lines within a bullet or step.

- In a DocBook procedure, one step can refer to a different step using a symbolic name. Internal step number references will be recomputed automatically.

For example, if step 7 says to go back to step 3, and then later someone adds a new step before step 3, this reference is now in step 8 and refers to step 4. MoinMoin won't handle this case automatically.

- If you use numbers like 1., 2., 3. to number a list, everything works fine. However, if you use step numbers a., b., c., ..., the indentation is incorrect: all steps after the first are indented an extra level. Try this test case:

```

a. This is the first step.

b. This is the second step.

c. This is the third step

```

The workaround is *always* to let MoinMoin do the numbering for you. If you number the steps a., a., a., ..., it works okay.

- Simple tables are easy to build in MoinMoin, but some operations are tedious. For example, if you have a 200-line table and you want each entry in a given column to be right-aligned, you must use the special “<)>” marker just after the opening “| |” on *each* line. DocBook tables allow you to describe the alignment of a column once initially, and that's all you need.
- Inside a verbatim section enclosed in “{{{...}}}", you can't get font changes. This makes it hard to show prototypes of program or file constructs, in which some of the parts are to be replaced. DocBook makes this easy: use `replaceable` elements inside a `programlisting`, and those elements will be shown in italic monospaced type.
- There is a minor bug in the mechanism for `WantedPages`. This procedure reproduces the bug.
 1. On some page A, add this macro to display links to all its subpages:

```
[[PageList(regex:^A/.*)]]
```

2. Create a page A/B. Revisit page A and it will appear in the output from the `PageList` macro.
3. Delete page A/B. Revisit page A and the new page will not appear.
4. Go to `WantedPages` and it still shows a dead link from A to A/B.

The workaround is to make some trivial edit on page A, such as adding a space at the end of a line somewhere. After saving the page, `WantedPages` will no longer show the spurious dead link report.

