Editing in XML — 1

Lou Burnard

The example files used in these exercises are all available on the web at `http://www.tei-c.org.uk/Talks/OUCS/2004-02/examples.zip`. Before starting work, you should download this file to your local disk and unpack it. Work on the H: drive on your PC.

1 Emacs exercise 1

There are literally dozens of different editors you can choose from to work with XML. For this course we will use the latest incarnation of a long-established favourite: GNU Emacs. The software is an attractive option for those wishing to produce high quality XML-encoded documents on a limited (or nonexistent) budget because it is genuinely ubiquitous, very widely understood, documented in great detail, and comes at a price that every academic project can afford (i.e. zero). It may not be as flashy as some XML editors, but it can still be configured to make entering valid XML text very simple. In this exercise, we'll use a version that has been customised to simplify data entry of TEI conformant documents of any kind. We won't try to explain all about emacs as there are dozens of books and web pages on the subject, and emacs also comes with its own built in help system and tutorial.

- Locate the emacs icon (it's a Gnu, since you ask) on your desktop and double click it.
- The emacs application window opens. Select Open File from the File menu, and navigate to the directory where you installed your sample files (H:/samples) for example. Find the file `verse.xml` and open it.
- At the bottom of the screen, below the status bar, is a small area known as the minibuffer which emacs uses to send you messages. When you open an XML file, emacs will automatically look for a schema file for it: the minibuffer tells you that it has found one in the file `verse.rnc`.
- If you want to find out about using emacs the traditional way (from the keyboard) you'll find more help than you can possibly digest on the Help menu. (Try the Tutorial for basics. We will be focussing on XML-specific features in the rest of this tutorial.)

1.1 Completing the tagging

The file you can see in the window uses colours to distinguish tags, attributes, and text. But the tags are just as easily edited as the rest of the text.

- Use the mouse or arrow keys to move the insertion point (the cursor position) anywhere in the document, and type or delete a couple of characters. What happens if you do this inside a tag?
- If you make the document invalid, by changing one of the tags, you will see the offending changes are highlighted (underlined in red). Also, the word `Invalid` will appear in the status line. In the nXML mode we are using, emacs constantly checks that your document is valid, and warns you when it is not.
- To undo the last change you made, select Undo from the Edit menu. If you made more than one change, you can correct it by retyping or deleting (most of the keys behave in the way you'd expect). Or you can simply close the file (choose Close on the File menu) and re-open it.

As you can see, we haven't done a very good job of tagging this poem: the last line actually contains several lines and stanzas run together. In this part of the exercise we'll try to improve on the tagging.

- Using the mouse or the arrow keys, put the cursor after 'upper lid;' in the third stanza. Type the two characters </. Observe what happens.
- Move the cursor to the start of the next line (before 'So I blew') and type `<l>`. Observe what happens.
- Now you need to repeat these two steps to add a `</l>`-tag at the end of this line, and a `<l>` tag at the start of the next one. Use the short-cut CTRL-E (for end) to move the cursor to the end of the current line. To get to the start of the next line, hit CTRL-F (for forward).
- When you have put the `</l>` tag at the end of the refrain 'And I still had a fly in my eye', type `</` again. Can you explain what happens?
- The sequence `</l>` closes the current element (the line). If you repeat it, you close the next element up the hierarchy (the line-group). Repeat it again to check you have understood. What tag must you insert at the start of the line 'And then Sir...'?

If you want a reminder of what tags are available, click on the file `tagging.html` in the samples directory.

1.2 Defining a macro

Computers are supposed to simplify boring repetitive jobs like typing in tags. Fortunately emacs has a built in macro facility which allows you to reduce the amount of typing needed for repetitive tasks like this. To use it you need to learn some rather arcane emacs keystrokes – but you will find the effort worthwhile.

In what follows, we use CTRL to stand for the control key and the hyphen to show that two keys are to be hit simultaneously rather than one after another. So, for example the sequence `CTRL-x (` means:

- Hold down the control key
- Hit the x key once; release both keys
- Immediately enter a (— on most keyboards this requires you to hold down the shift key, of course

Defining a simple macro is like making a recording of your keystrokes, which you can then playback as often as you like. In this part of the exercise we'll define a macro which just repeats the sequence you practiced earlier – putting a start-tag at the start of a verse line and an end-tag at its end.

- Put the cursor at the start of the next verse line. ('And then Sir...'). It's important to start in the right place when defining a macro!
- Type `CTRL-x (`. The message `Defining kbd macro...` appears in the minibuffer. From now on, whatever you type will be recorded as part of the macro.
- Type `<l>`. Type CTRL-E to move to the end of the line. Type `</` to finish the line, and then CTRL-f to move forward to the start of the next line. It's important that a macro winds up in the right place.
- Type `CTRL-x )`. The message `Keyboard macro defined` appears in the minibuffer to confirm that your macro is now available.
- Type `CTRL-x e` to execute your keyboard macro. Did it work as expected? If not, you will need to redefine it, having cleared up any mistakes it introduced...

If your macro works as you expect, you can now use it to help with the rest of the tagging. If typing `CTRL-x e` once per line seems like a bore, you can tell emacs to repeat the command a specific number of times for you. For example, you could type `ESC 4 CTRL-x e` to execute your macro 4 times.

1.3 Adding more tags

Some of the verse lines in your example actually take up more than one typographic line. The tag `<lb/>` could be inserted at the point where a linebreak occurs, if you want to mark this fact. The poem also contains several emphasised words (they are in italic, in the original printed version), which you might want to tag using the `<emph>` tag.

To find out what tags are valid at any point in the document, you can use the emacs completion feature. Proceed as follows:

- put the cursor in front of the emphatic 'I've' in the refrain of the second stanza and type <
- a message appears in the minibuffer to warn you that `<I've>` is an Unknown element.
- Hold down the CTRL key and press the RETURN key. A new window opens which (amongst other things) lists the tag names which are valid at this point in your document. The text in the minibuffer changes to read `Tag:`
- You can select the tag you want by typing its name into the minibuffer, or by right-clicking on its name in the list. Press RETURN to add the chosen tag to your document, and add its closing >.
- To move the cursor to the end of a word, type `CTRL-right arrow`. Type `</` to add the appropriate end-tag as before. Now tag the remaining `<emph>` elements in the document.
- If you want to add linebreaks, remember that this is an empty element: you can type the whole thing in one go: as `<lb/>`
- If you want to add an attribute to any element, type a space after its name, and press `CTRL RETURN` as before. If the element only has one attribute available (`<lb>` for example), emacs will insert it as `n="`. Type in the number, the closing quote and the closing `/>`. If the element has more than one attribute (`<div>` for example), a list of the available attribute is displayed, for you to select from, as with tags.

1.4 Adding character entities

The poem contains a number of dashes, which have been represented as double hyphens. The dash is a normal Unicode character, but like accented letters and other special punctuation marks is difficult to enter from the keyboard. Emacs has two ways of helping you:

- Put the cursor in front of the two hyphens used to represent a dash in the text. Select the UniChar menu and scan the list of available character names for the one you want: mdash.
- Emacs inserts the correct character for you. You can now delete the two hyphens.
- Now that you know the name for this character, you can also type it in directly. Move to the next two hyphens and this time type the characters `&mdash;`. When you type the semicolon, the required character appears.

The second method is very useful for accented letters. The character names emacs recognizes in this way are the same as the entity names used in HTML and other SGML derivatives: for example, to enter an e with an acute accent, you could type `&eacute;`.

1.5 Global tidying up

While we are looking at semicolons, you may have noticed that this document has redundant whitespace in front of them. We will use this as an excuse to introduce you to one of emacs's more powerful and useful features: the conditional search and replace. Most editors have some kind of search and replace: emacs has several.

- Choose Search from the Edit menu, and then Replace from the submenu which opens from it. Alternatively, type `ESC %`.
- The cursor moves to the minibuffer, where the prompt `Query replace:` appears. Type the string you want to replace, in our case ` ;` (space followed by a semicolon), and press RETURN.
- The word `with` appears in the minibuffer. Type the string you want to see instead, in our case `;` (just a plain semicolon please), and press RETURN again.
- If the string you typed in is present in the file, the cursor will move to it, and highlight it. The message in the minibuffer now says `Query replacing ; with ; (? for help)`. If you're sure you want to replace all occurrences in the file, just type `!`. If you want to replace this one and then move on to check the next, type `Y`. Type a question mark to see what other options you have.

You may also want to experiment with the cut and paste options on the Edit menu. These work in much the same way as other editors, though few editors have quite so many of them.

1.6 And finally...

Don't forget to save your work when you have finished! Use the Save command from the File menu, or the Save As command if you want to save it under a different name.

In conclusion, consider the following:

- How many things did you want to tag in this text for which no tag was defined by this simple schema?
- Do you think any of the rules in this schema seemed inappropriate for this kind of text?

2 Exercise 2

In your samples folder, you will find versions of the other parts of the Punch page called `cartoon.xmlplay.xml` and `paras.xml`. Each has been given just enough tagging to be well formed XML, but no more.

Open each in turn with emacs and bring its tagging up to spec. For these files, you will use a much richer schema, called teixlite.