Toward program development and documentation tools

It has been a while since the CASIO FX702P's BASIC and the IBM 360 Fortran punch-cards. Throughout all these years programmers would every once in a while voice weak complaints about being under-tooled for their complex jobs. We are all extremely busy to get done with the current application, and documenting is traditionally a long-procrastinated chore. The net result is that programming has often the flavor of black magic, and making sure that a piece of code is actually living up to its meant-to-be purpose is an activity bearing a lot of common traits with expecting to win the lottery...

Reading one's own or worse-someone else's, code is often times a task without fun or gratification, since things such as indentation are not 100% consistent even among the most fervent adepts of the latter.

Let us go to the simple example of how hard it is for both a Matlab novice and a seasoned user alike to find which exactly 'end' is missing or too much, and why.

Let us start from the Xemacs window on the left displaying a snippet of Matlab code where just one 'end' is missing.

Every serious developer knows about emacs and its 22 versions todate. No one is indifferent: you either love it, or you hate it…

Those who love it swear by it and consider it the top of code editing and development tools.

As such, it is likely to provide us with a nice baseline for comparisons.

Let us start by ease of use. Though, I have a bit of resentment to its elegant and verbose lisp constructs, I am not exactly new to emacs and so I installed Xemacs version 21.4.21 on my Xppro laptop.

Then Matlab central gave me a flying start toward:

HOWTO: get matlab mode working with xemacs on windows:

http://www.mathworks.de/matlabcentral/newsreader/view_thread/159508

Even so, it took me close to a whole afternoon to actually get Xemacs to highlight Matlab syntax.

Frankly, I think that an average user of Matlab (but not emacs) would not even get halfway through the process without 100% reliable guidance.

And unfortunately the web abounds with stuff like here:

> http://www.cs.ait.ac.th/~mdailey/matlab/
>
> …I don't pretend to understand any of this emacs stuff. For more information, you might try the Mathworks Matlab Central Contrib site. (i.e. back to EXACTLY where we started from).

And then the poor novice starts going to rounds of (emacs-lisp-hell) like:

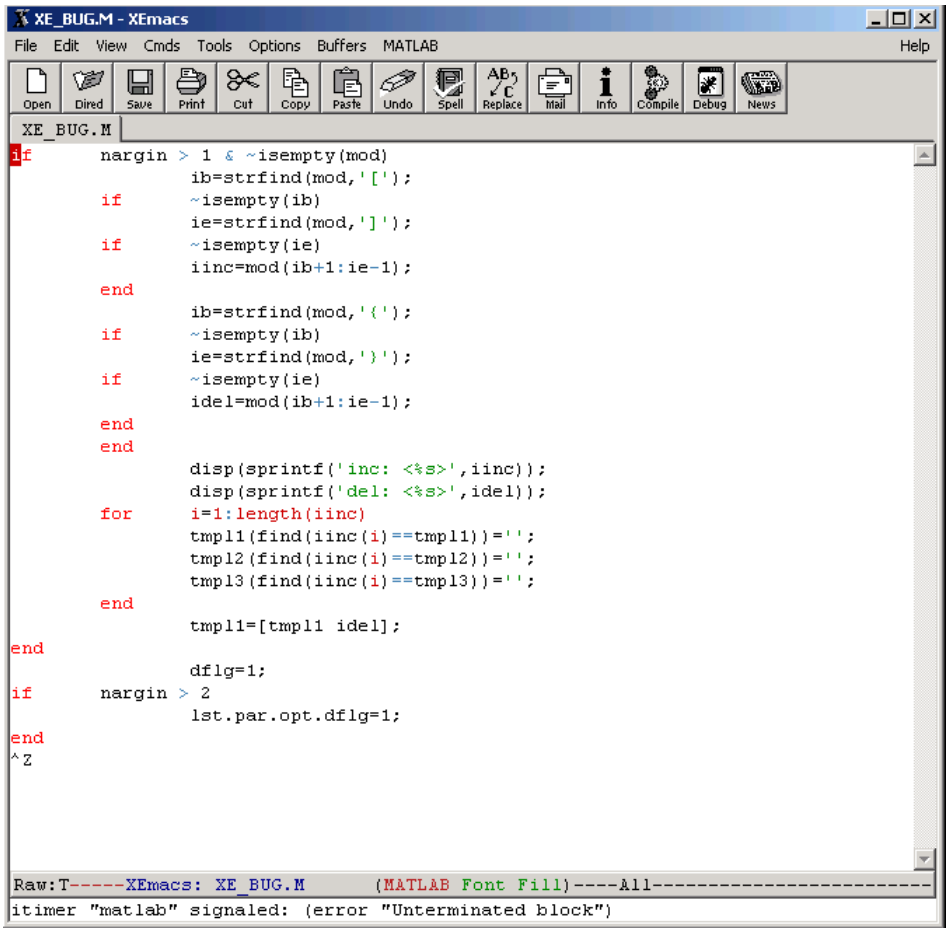
> ***what is the emacs exact load path ?***
>
> and
>
> ***how come neither HOME, nor even .xemacs is on it?***
>
> http://www.rattlesnake.com/intro/Loading-Files.html
>
> http://www.emacswiki.org/emacs/LoadPath

and even after finally succeeding to load 'matlab.el' :

> ***why Xemacs will still not highlight Matlab syntax?***
>
> http://vis-www.cs.umass.edu/~mmattar/teaching/vision-spring06/.emacs
>
> ;; Syntax highlighting
>
> ;; Unfortunately emacs and xemacs require different commands for this

```
;; Comment/Restore one of the two commands below

;; Syntax highlighting for *emacs*

(global-font-lock-mode t)

;; Syntax highlighting for *xemacs*

;; (font-lock-mode 1)
```

Ok. Now, finally we see syntax highlights, and there is a long list of 275 (!) Matlab-related 'macros' accessible from within Xemacs. How useful is all this for debugging, or are we remaining thirsty sitting by the ocean of salty water?



Exactly how useful is this Xemacs message full of apparent wisdom:

"Unterminated block"?

How well aware does the **uniformly red** Xemacs highlighting make us of the intended logic of our code?

What if 2 or 3 **end**'s were missing?

Conversely the following is achieved using very native Matlab, and does make us more aware of the discrepancy to the intended logic of our code.

```
0001 if      nargin > 1 & ~isempty(mod)
0002                 ib=strfind(mod,'[');
0003         if      ~isempty(ib)
0004                 ie=strfind(mod,']');
0005         if      ~isempty(ie)
0006                 iinc=mod(ib+1:ie-1);
0007         end
0008                 ib=strfind(mod,'{');
0009         if      ~isempty(ib)
0010                 ie=strfind(mod,'}');
0011         if      ~isempty(ie)
0012                 idel=mod(ib+1:ie-1);
0013         end
0014         end
0015                 disp(sprintf('inc: <%s>',iinc));
0016                 disp(sprintf('del: <%s>',idel));
0017         for     i=1:length(iinc)
0018                 tmpl1(find(iinc(i)==tmpl1))='';
0019                 tmpl2(find(iinc(i)==tmpl2))='';
0020                 tmpl3(find(iinc(i)==tmpl3))='';
0021         end
0022                 tmpl1=[tmpl1 idel];
0023 end
0024                 dflg=1;
0025 if      nargin > 2
0026                 lst.par.opt.dflg=1;
0027 end
```

```
0001 if      nargin > 1 & ~isempty(mod)
0002                 ib=strfind(mod,'[');
0003         if      ~isempty(ib)
0004                 ie=strfind(mod,']');
0005         if      ~isempty(ie)
0006                 iinc=mod(ib+1:ie-1);
0007         end
0008         end
0009                 ib=strfind(mod,'{');
0010         if      ~isempty(ib)
0011                 ie=strfind(mod,'}');
0012         if      ~isempty(ie)
0013                 idel=mod(ib+1:ie-1);
0014         end
0015         end
0016                 disp(sprintf('inc: <%s>',iinc));
0017                 disp(sprintf('del: <%s>',idel));
0018         for     i=1:length(iinc)
0019                 tmpl1(find(iinc(i)==tmpl1))='';
0020                 tmpl2(find(iinc(i)==tmpl2))='';
0021                 tmpl3(find(iinc(i)==tmpl3))='';
0022         end
0023                 tmpl1=[tmpl1 idel];
0024 end
0025                 dflg=1;
0026 if      nargin > 2
0027                 lst.par.opt.dflg=1;
0028 end
```

Finally, there are many more tools like this one (some already on File Exchange) that if well packaged and streamlined can provide a sweet and significant difference for Matlab developers.

Here I don't provide any code (yet), but I'd like to hear your voice of feedback about what you fellow Matlab developers think.

The more you express enthusiasm, the sooner there will be actual code available for download.

The more you criticize and the more constructive suggestions and comments you post – the better it will be.

Here's a free idea: How about a program-development blockset? E.g. A bit like the LabView stuff?

And yes, I know about 'Stateflow', but it is not quite the same. Or is it?

Which do you think are among the best (most useful) submissions to File Exchange that should make it into such a toolbox.

I have found a couple

e.g. M2HTML: http://www.mathworks.com/matlabcentral/fileexchange/4039

It is believed that now is the time to turn the power of Matlab onto itself, closing yet another of Hofstadter's 'strange loops', i.e. provide actual tools and functional help to the Matlab developers to address the fundamental challenges of their profession by 'packaging' into a toolbox of a set of ideas and codes. These are the product of a wide and international community of Matlab developers, but have been under-exploited and sporadic todate.

References:

Matlab file exchange

en.wikipedia.org/wiki/Strange_loop

(To be continued.)