Subject: Re: IDLdoc 3.3.1 bug fix release
Posted by Marc Buie on Tue, 11 Jan 2011 20:57:20 GMT
View Forum Message <> Reply to Message

Mike -

I've tried your package once before but put it aside. Stimulated by your newest post, I tried your latest today. Sadly, it still doesn't work very well for me. I hope you don't mind but I thought I'd share some of my experiences here. Perhaps it will provide some inspiration.

I have a very large, and old IDl library. Some of it's actually good code, some is perhaps not as good. But, this is a collection of 500+ routines and there's going to be a lot of variety.

Way back in 1990 when I wrote my first IDL routine I used the document header template from the Astronomy User's Library (unchanged to this day, I think). During my first year of coding I found a couple of minor deficiencies in the document header. The biggest of which was the need to distinguish between keyword input and keyword output parameters. I have been pretty good about writing documentation, at least for the more important routines, but all of this is based on my own personal variant of the document header (which of course traces its origins back to the one put forth by IDL in even earlier days).

As nice as the IDLdoc and rst document forms might be, there is essentially zero chance that I will rewrite all my documentation to conform to a new "standard". You package does indeed support the original form of the IDL document header but it places a number of restrictions on the _content_. As a result, I don't get a very satisfactory output from your idldoc tool (it doesn't even build the index.html file). Of course, I'm violating all manner of rules in formatting the header and the software can't be expected to work in this case.

What I wonder is if you can afford to lighten up on the syntactical requirements of the ancient document header and still get something useful. There's really no way to get the full rich-ness of the documentation experience out if I don't adopt and use "rst" but surely there's some middle ground.

I've written my own parsers and decided the only thing I can count on is a line of the form:

; TOKEN STRING:

where "TOKEN STRING" is just some descriptive (and short) string that labels a section. You might have more than the "standard", you might have less.

For my own limited documentation tools I further required a minimum set to exist: NAME, PURPOSE, and CATEGORY. One additional restriction I impose is that PURPOSE should not be longer than one line. If these three exist then I can do something sensible with building documentation. The rest of the fields can be there, or not, and you'll get documentation that is at least no worse than the document header itself.

For reference, this is my document header template:

```
;+
; NAME:
; PURPOSE:   (one line only)
; DESCRIPTION:
; CATEGORY:
; CALLING SEQUENCE:
; INPUTS:
; OPTIONAL INPUT PARAMETERS:
; KEYWORD INPUT PARAMETERS:
; OUTPUTS:
; KEYWORD OUTPUT PARAMETERS:
; COMMON BLOCKS:
; SIDE EFFECTS:
; RESTRICTIONS:
; PROCEDURE:
; MODIFICATION HISTORY:
;-
```

By the way, I found it rather amusing to look at the McCabe complexity measure on some of my code. I'd never heard of this before but it's cool to see the results. I didn't see any real surprises, though.

Anyway, as with many of us that write enough code, we have to find ways to handle the documentation along the way. I've got my system and it works well enough. But, I do see a lot of great functionality in your system that would be very helpful with my large library. There's no chance I'm going to invest time into something as sophisticated as your "rst" system but I'm unfortunately locked out of any other benefits because of the syntax requirements and the large cost to convert.

Not sure what the final answer is here, but, hopefully you find some value in these thoughts.

Cheers,
Marc