The Fox Hunt

By Whil Hentzen

[Darren, you can include my CIS ID in my bio for future articles: 70651,2270.]

Commenting one's code is the bane of virtually every programmer, giving rise to the oft-repeated line - "Of course I don't document my code. If it was hard to write, it should be hard to understand." Nonetheless, as professionals we recognize the need to do so. I've seen many types of program and function headers and no two are alike. Over the years, I've decided on a style that is fairly painless to produce but provides all the information I need.

The first essential piece of information in a program header is the name of the file. Seems silly to point it out, but I wish I had a nickel for every PRG file I've seen without an identifier. These were often short programs that were going to be used temporarily, but then they grew...

```
*     Program: AB.PRG (Array Browser)
```

The next essential line is the purpose of the program. Not how it works - just what it does. Along with this, I include the versions of FoxPro for which it was designed Sounds trivial, but it sure helps when you come across a program with commands that you've never seen before - until you realize that this was an old FoxBase procedure that was never given a proper burial.

```
*     Purpose: Display the contents of an array in a Browse format
*   Platforms: FPD/W 2.5a
```

The third essential part of a program header is the calling syntax. First, show the generic syntax. I use a different style than Microsoft does when showing parameters:

```
*      Syntax: do AB with <c ArrayName> [, <n ArrayDimension>]
```

Note that the data type is part of the parameter syntax, so I know what type of data the program is expecting. Then I describe each parameter, like so.

```
*       Parms: <c ArrayName> text string of array name
*            : <n ArrayDimension> if <c ArrayName> doesn't exist, AB will
*            :  display a list of arrays in memory - this parameter can be used
*            :  to limit the list of arrays
*            :  0 - list both 1 and 2 dimensional arrays
*            :  1 - list only 1 dimensional arrays
*            :  2 - list only 2 dimensional arrays
```

I don't need to explicitly include the data type in the parameter description since it is covered in the parameter syntax.

Finally, I provide several "real-life" examples. Doesn't it bug you to see a command in the Language Reference with four or five possible parameters, but only one example? You can spend hours experimenting with various combinations of parameter types and syntax until you figure out the peculiarities of a particular command. I try to provide enough examples to "exercise" each parameter.

```
*     Samples: =ab("aDealers")
*            : do AB with "aDealers", 2
```

Here, I needed only two examples to show the program being called with one or with two parameters, and to show how it could be called as a program or as a function.

The fourth essential part of a program header describes the relationship the program has on the environment. Specifically, does it require any files to exist in order to run, and does it create any files that are (intentionally) left behind? We've all seen programs that crash because of "XXX.DBF not found" errors, and we've all seen directories filled with hundreds of temporary files that seem to be no longer needed, but that you don't dare delete, "just in case."

```
*       Files: None
*    Required:
*            :
*       Files: ZYXWVUTS.DBF is created and left on disk for later reference.
*     Created: If array name is not found, ZYXWVUTS.RID & ZYXWVUTS.TXT are created
*            : and both are deleted if the function terminates normally
```

You can create a “dummy” header that you can pull into new programs, but you may find it faster, and more “foolproof” to name the file that contains this blank header UNTITLED.PRG and place a copy in your default directory. Then, whenever you issue the command

```
modify command
```

(without a file name or question mark), the UNTITLED.PRG file will be brought up automatically.

Another trick you may find useful is to use special character combinations before certain comment lines, and then write a program that will scan each PRG and pull just those lines into a single reference file. For instance, by marking the file name and purpose of a PRG like so:

```
*(    Program: AB.PRG (Array Browser)
*(    Purpose: Display the contents of an array in a Browse format
```

you could search for lines beginning with “*(“ and quickly produce a reference list of all programs and what they do.

I’ve found that these four pieces of information help me focus on the purpose of the program and make sure I don’t run down too many dead ends. You may choose to use your own, but the important point is to make your commenting short and simple - overly complex schemes are less likely to be used - and even less likely to be updated as changes are made to the program.

[Darren, I’ve included a complete program header in the event that you’d like to use it - sometimes it’s hard to find something to fill that last 2/3 of a column...<g>]

```
*     Program: AB.PRG (Array Browser)
*             :
*   Architect: Whil Hentzen
*     Contact: Hentzenwerke
*             : PO Box 17343
*             : Milwaukee WI 53217-0343
*             : U.S.A.
*             : Voice: 414.332.9876
*             : Fax: 414.963.4999
*             : CompuServe: 70651,2270
*             :
*   Copyright: (c) 1994 Hentzenwerke
*             :
*             : This program is provided "as is" without warranty
*             : of any kind, expressed or implied.  IN NO EVENT
*             : SHALL ITS AUTHORS, CONTRIBUTORS, OR DISTRIBUTORS
*             : BE LIABLE FOR ANY COMMERCIAL, SPECIAL, INCIDENTAL,
*             : OR CONSEQUENTIAL DAMAGES.
*             :
*   Hierarchy: Standalone function
*             :
*     Program: AB.PRG (Array Browser)
*     Purpose: Display the contents of an array in a Browse format
*             :
*     Version: 1.0
*             :
*   Platforms: FPD/W 2.5a
*             :
*      Syntax: do AB with <c ArrayName> [, <n ArrayDimension>]
*             :
*       Parms: <c ArrayName> text string of array name
*             : <n ArrayDimension> if <c ArrayName> doesn’t exist, AB will
*             :  display a list of arrays in memory - this parameter can be used
*             :  to limit the list of arrays
*             :  0 - list both 1 and 2 dimensional arrays
*             :  1 - list only 1 dimensional arrays
*             :  2 - list only 2 dimensional arrays
*             :
*     Samples: =ab("aDealers")
*             : do AB with "aDealers", 2
*             :
*      Return: nothing
*             :
*       Files: None
*    Required:
*             :
```

```
*       Files: ZYXWVUTS.DBF is created and left on disk for later reference.
*     Created: If array name is not found, ZYXWVUTS.RID & ZYXWVUTS.TXT are created
*            : and both are deleted if the function terminates normally
*            :
*       Notes: Function returns error message if no parm is passed
*            :
*            : Function looks for arrays in memory (via DISP MEMO)
*            : if parm passed is not an array and presents a list
*            : of available arrays choose from
*            :
*      Hidden: None
*   Knowledge:
*            :
*         ERs: Write edits to browse window back to array
*            : Filter parm for browse window
*            : Keep browser open & updated during program execution
*            : Enable sysmenu
*            : Browse two arrays at same time
*            :
```