[Selectively include a FoxPro Command Window in your applications with RunFox]

Last year we looked at a couple of tools to give you a "portable dot prompt" - the ability to run interactive FoxPro commands without having the full version of Fox installed on the machine. This capability is handy when you've installed an application using the Distribution Kit instead of requiring your users to have FoxPro themselves, but you have the need to travel outside of the realm of the application.

Since then, I've been made aware of a utility that not only provides this same ability but further enables you to selectively include a Command Window in compiled applications. Ed Leafe's RunFox utility displays a mock Command Window in which most regular FoxPro commands can be run. RunFox can be compiled as a standalone EXE or as a compact EXE that runs in conjunction with FoxPro's runtime libraries. You can also call it from a menu option inside your application.

Since we've already covered the uses of a "portable dot prompt" in previous articles, this column will focus on using RunFox from within an application. You can make RunFox available to you in one of two ways from within an app: via a hot-key, or via a menu choice.

Going the hot-key route is as simple as including the line

on key label ALT-F12 do RUNFOX with .T., .T.

somewhere in your application's setup code. The same syntax can be used in a Command fired off by a menu choice.

Oftentimes, however, you might not want your users to have access to RunFox. You could use a "secret" hot key using some arcane combination of keystrokes (Shift-Alt-Backspace), but that's only good for as long as you can use it without your user looking over your shoulder.

I include several "programmer-only" menu options on my standard Help popup, such as Debug and Trace. They are only visible when I log onto the system with my password, courtesy of the magic of GENMENUX (see Foxtalk, March 1994). RunFox is one more menu option that belongs to this class.

*\\\ screenshot COOLRF01.BMP goes here
Figure 1: Include RunFox on the menu as one of several "developer-only" menu bars.

Using RunFox

Once RunFox is executed (we're inside an application, remember), the Command Window displays and the application's menu is replaced with the standrad FoxPro sustem menu. You can now execute a FoxPro command just as you would within FoxPro itself. Editing and executing previous commands, using the semi-colon for command continuation, and pressing <ESCAPE> before executing a command in order to cancel it all work like they do within FoxPro. Furthermore, strings that FoxPro does not recognize (say, typos) will issue an error instead of crashing the app. Commands that are not appropriate for the Command Window (say, DO CASE) are also trapped.

However, since RunFox's Command Window is actually just an Edit field built with the screen builder, there are a few small differences between using RunFox and FoxPro's Command Window. RunFox parses the contents of the Command Window; after entering a number of commands, you may find the response time becomes sluggish. Entering the command "CLEAR CMD" will clear the window and enable you to start fresh.

You can also press F2 to mimic the Ctrl-Alt-Shift action in FoxPro that hides and displays all windows in order to see the contents of the desktop.

To exit RunFox and return to your application, issue any of the commands: QUIT, EXIT, CANCEL or type Alt-Q.

Special notes about variables

When you create a variable within FoxPro's Command Window, it's is automatically created as PUBLIC, which means it is available everywhere. However, since RunFox's Command Window is actually an SPR that is called by a PRG, any variable

created within the Command Window is actually created in a procedure called by the SPR. As a result, that variable is local to the procedure and becomes invisible to the rest of the system as soon as the procedure is exited.

For example, suppose you want to SEEK in a table for an expression. If you issue the commands

```
use MUSICIANS
m.cSeekString = “Madonna”
seek m.cSeekString
```

the SEEK command will blow up, and not just because the computer has choked at the thought of Madonna being grouped in with “Musicians.” Rather, the variable m.cSeekString will not be available to the SEEK command because it was local to the procedure that created it in the previous command.

*\\\ screenshot COOLRF02.BMP goes here
Figure 2: Executing RunFox changes the menu and displays a Command Window. Memvars not decalred PUBLIC in the RunFox Command Window are private to a procedure within the RunFox Command Window .SPR.

In order to make a variable available throughout your RunFox session, declare it PUBLIC within the RunFox Command Window. RunFox will keep track of the variables you declare PUBLIC, and will release them when you exit RunFox. The syntax you’d need in the above example is

```
PUBLIC m.cSeekString
m.cSeekString = “Madonna”
```

*\\\ screenshot COOLRF03.BMP goes here
Figure 3: Declaring memvars PUBLIC ensures you can use them in subsequent commands.

RunFox automatically declares a set of 26 variables, A through Z, to be public, so that you can issue “quickie” commands like

```
count for nCost > 100 to A
```

and have the result remain public for the rest of the RunFox session.

A caveat when using arrays in RunFox: Do not declare an array PUBLIC and dimension it in the same statement. The procedure that actually parses the commands you type is not set up to handle a PUBLIC command that also contains array subscripts. Use two commands:

```
public aRockNRoll
decl aRockNRoll[10,10]
```

Under the hood

At first glance, this may seem to be a trivial undertaking, but a peek under the hood shows the care and depth that has gone into this tool. Since it’s designed to be run within an application, RunFox has to take particular care not to step on the environment, and to restore everything just as it was when called. As a result, it can save the current View (tables and selected work area), the screen, the menu, and various other settings.

The two Logical parameters that you pass with the DO RUNFOX command are switches that tell RunFox whether or not to save the View and the Screen. Take a moment to look through the code - the source is clearly laid out and well-documented, and there are a number of tips and hints.

Where to get it

The RunFox package (a ZIPPED file named RUNFOX.ZIP) consists of about a dozen files - the RunFox project, Command Window screen set, several supporting programs, and documentation files. Now that you’re convinved that RunFox is one of those "gotta haves," remember that this is shareware - if you like it and use it, send the author $13. You can find it on this

month's Companion Disk or in CompuServe's FoxForum libraries. You can contact the author, Ed Leafe, on CompuServe at 72530,1175.