Session E-LIN2

Technical jumpstart how VFP works with Linux, Part 1

Whil Hentzen

Contents

When you are asked to connect your VFP app to a Linux back end this year or next – and you will be – you can either offer them a solution, or a referral to someone else who had the foresight to prepare in advance and now knows something you don't.

In order for you to be able to work with a Linux back end, you're going to need to know something about how Linux works, and the best way involves a two step process. First, plunk down a Linux workstation on your desk next to your Windows machine and develop some experience with the new OS. Second, once you have a basic level of comfort with Linux, gained through your experience on a workstation, leverage that knowledge and learn to connect to a Linux server from your Windows machine.

This session shows you both of these processes. What to expect when you go about setting up your Linux workstation, how to set it up, how to connect to your Windows network, how to fit VFP into the mix, and even how you might use it to replace your Windows workstation in some cases. And then how to connect to an existing Linux server, running MySQL or another back end, and then get your VFP apps talking to that back end data.

This session is the meat and potatoes sequel to E-LIN1. You'll learn the nuts and bolts “how to” procedures to get up to speed with a Linux workstation and connecting to a Linux server.

Installing Linux on your Workstation

This section is designed for the Visual FoxPro developer who hasn't ever installed Linux before.

1. Obtain a copy of Linux.

Your first step is to get a copy of a Linux distribution. The popular distributions include Red Hat (www.redhat.com), SuSE (www.suse.com), and Mandrake (www.mandrake.com). I'll refer to Red Hat during this paper since it's the one I have the most experience with.

The current Red Hat distribution is version 9.0 and is available on CD or via download. The CD package is available for about $100 US, and includes about 8 CDs, including a DVD. The first three, labeled “Installation – Disk 1/2/3” are the only ones you're concerned with.

If you're going to download, go to www.redhat.com/download/mirror.html and select a site that has the Red Hat logo under the “i” column, for i386 type processors, and that has a link for “Distribution”. Once you click on a site, select the .download vehicle you want to use – HTML, FTP or RSYNC. Sites have a variety of interfaces – some simply provide a folder drill down while others have nicely formatted lists of links. It can be confusing to navigate through to find the files you want, especially because it may not be obvious which files you're after.

Look for links to files named like so: shrike-i386-disc1.iso (not shrike-rpms-disc1.iso). The 'shrike' represents the code name for Red Hat version 9 (version 8's name was “Psyche”, for example.) The “disc1” clause represents disk 1, obviously. You'll want three files, one each for disk 1, 2, and 3.

They'll each be about 600 MB, so make sure you've got disk space and a fast connection.

These are ISOs – disk images – use your CD burning software to copy these ISOs onto the disks. Don't do a file copy – you want to create an image on the CD.

Once you've got three CDs, either via purchase or by downloading, it's time to begin the install.

2. Decide on whether you want to do a Linux-only installation or install Linux alongside another operating system.

If you want to do a Linux-only installation, copy the data on the target machine somewhere else, because you'll be formatting your entire hard disk.

If you want to install Linux along side another operating system, which is called a dual-boot, you'll want to do two things. First, back up anything of value on the target machine. Second, ensure there's enough room on the hard disk of the target machine to put Linux. You'll need a minimum of one gigabyte, but you'll be much happier if you have between three and five gigabytes. How much you have available will determine how much software you can install.

After you've freed up enough space on the target machine, you'll need to make sure that Linux can take advantage of it. There's a difference between “unused space” and “free space”. The former is simply space in a formatted partition that isn't being used, while the latter is space that is not formatted and/or part of a partition. Once you've got unused space in a partition, you may need to resize it (using something like “Partition Magic”) in order to create free space.

For example, suppose you've got a 20 GB hard disk with two partitions. The C partition is 6 GB and the D partition is 14 GB. Supposing that you want five GB free, you would first clear out enough space on the D partition so that there's at least 5 GB empty (unused.) Next, you would resize the D partition so that it was now 9 GB in size. At this point, you would have a 6 GB C partition, a 9 GB D partition, and 5 GB of free space. You'll put Linux in that 5 GB of free space.

Did I mention that you should back up your data?

3. Boot into the installation program, via CD or floppy

Most modern machines can boot from CD. If yours can, the first Linux CD is bootable, and will launch you into the first installation screen. If you can't boot from your CD drive, there's a file on the first CD (see the README for details) that describes how to create (ugh!) a boot floppy.

In either case, you'll soon be staring at the installation screen. If there's another operating system on the target machine, you'll be offered the choice of whether to wipe it out and install Linux all by itself, or to install Linux alongside the existing operating system.

The next few screens are self-explanatory for anyone who has installed Windows more than about once. The first screen welcomes you to the installation process via a two pane window. The right pane has the welcome message and the left pane has help topics.

The next screens allow you to choose the language you want to use for the installation, the keyboard configuration, and the mouse configuration. If there's a previous version of Linux on the target machine, an "upgrade or install" screen automatically displays.

The next screen asks which type of installation you want to perform – desktop, workstation, server, or notebook. The Red Hat Linux distribution contains hundreds of software programs and utilities – all the way from personal productivity utilities that would be useful on a desktop to server programs like Web servers, email servers, and file/print servers. Depending on what type of machine you're building, you'll want to install some of these programs but not necessarily others. For example, you would probably want OpenOffice.org on a desktop machine but not on a Web server, while you'd probably want a DNS server on a Web server machine, but not on a desktop machine.

You'll have a chance to select which programs you want to install during the installation. However, since that list is long, and thus, rather daunting, the installation type you choose simply preselects groups of programs that are appropriate for the type you want. The next screen will allow you to customize that list of programs – for example, you may decide to add Engineering and Scientific tools to a desktop installation.

If you're undecided about what type of installation to choose, pick Desktop or Workstation and then examine the choices that are preselected for you in the next screen. You can go back and forth and change your mind if you like.

4. Decide on partition sizes

The next screen asks you about how you want the machine's hard disk partitioned. You can choose to have the installation program do the partitioning for you or to do it yourself via Red Hat's Disk Druid partitioning tool. If you choose automatic partitioning, you'll need to choose whether you want to remove just the Linux partitions, remove all of the partitions, or just create a Linux partition on the existing free space. This third option, of course, presumes that you have enough free space for the installation.

You can select Automatic partitioning and then click the Review button to see what choices have been made, and then manually modify those choices with Disk Druid.

Did I mention that you should back up your data before you partition?

You may be nervous about choosing manual partitioning with Disk Druid but you shouldn't be. Unlike the inscrutable FDISK program from the Windows world, Disk Druid has a reasonably intuitive and safe interface. The only trick is to know which settings to choose. And that's why you're reading this.

If you choose to partition manually, you need to know a little bit of how drives and partitions are named, and what goes on a partition. I'll cover that first, and then provide a sample partitioning that you can use if your machine's configuration matches the example.

Linux names IDE drives with the letters "HD" and then a letter of the alphabet. A computer with three disk drives would have them named HDA, HDB, and HDC. Within each physical drive, Linux names partitions with

a number appended to the hard disk name. So if the first hard disk has two partitions, the second has one partition, and the third has four partitions, you'd see drives and partitions named

HDA1, HDA2

HDB1

HDC1, HDC2, HDC3, HDC4

Linux doesn't have arbitrary names for drives like “Drive C” and “Drive D”. Everything on the collection of Linux hard disks is contained in a single hierarchiel folder structure like you'd find on a single Windows drive. Linux associates specific directories in that hierarchy with specific partitions “behind the scenes”, but that's transparent to the user. For example, suppose a Linux filesystem hierarchy looked like this

/

/boot

/etc

/home

/usr

/tmp

All of these directories could be located on the same partition. However, you could also create several partitions, and instruct Linux to put everything except /boot and /home on the first partition. Then you'd tell Linux to put the contents of /boot on partition number 2, and /home's contents on the third partition. When you're working with the directory structure, you would navigate through the folders as if they're all on one partition on one drive. This idea takes a bit of getting used to, and earlier versions of Linux didn't make it easy to understand what was happening. Once you do one or two installs, it'll all make sense.

So, how many partitions should you create with Disk Druid, and what are they used for? The answer is four. The first is for /boot, the second is for /, the third is for a swap file, and the fourth is for /home. In essence, the entire folder structure will reside on the second partition, with the exception of /boot, /home, and a swap file.

This may seem like a lot of work to no apparent benefit. Why bother? The answer is that you can configure each partition with different rights and permissions – for example, the /boot partition – the one that contains all of the programs needed for starting up the machine - could be mounted “read-only” so that no one can accidentally (or intentionally/maliciously) mess with it.

Each of the directories that is associated with a separate partition is called a “mount point”. “Mounting” a directory is simply the Linux way of saying associating a directory (or a physical device, like a CD-ROM drive) with a partition. Thus, /boot is mounted on one partition, /home is mounted on a second partition, and so on. I bring this up because the swap partition isn't a mount point like the others. Instead, it's actually a different type of file system. An example will make it clearer.

In Disk Druid, here are the settings you'd use for these four partitions on a drive, in the order you'd create them:

Mount Point	File System Type	Size
/boot	ext3	100 MB
/	ext3	5000 MB
(none)	swap	1000 MB
/home	ext3	(rest of available space)

The sizes are suggestions for a machine with 512 MB of RAM and, say, a 20 GB drive. The rules of thumb are that (1) /boot only needs 100 MB (you could go down to 50-75 MB if you were really cramped for space), (2) / should be a minimum of 2 – 4 GB if you have a small hard disk (under 10 GB), and on up to 20 GB if you've got 60 GB or more, (3) swap should be twice the amount of RAM in the machine, and (4) /home should take the rest of the disk.

As you gain experience, you may decide to modify these to suit the preferences you develop. For example, some people put /etc (configuration files) on its own partition, and others put /var and /tmp (variable sized files like mail and print spools, and temporary files) on their own partition as well.

By the way, you've probably noticed that the Linux filesystem uses forward slashes, not backslashes like Windows. How do you remember which is which? I remind myself that Windows is holding me back, while Linux is a big step forward.

5. Choose boot loader

The next step is to decide on the boot loader. The boot loader is the program that tells the machine how to start up and where to find the /boot partition, the /boot partition, and where the programs are. Linux has two widely used boot loaders – LILO and GRUB. For your first installation, go with the default choice of GRUB.

6. Configure network settings

The next screen asks you to configure your network settings. These are similar to Windows settings. Your choices will depend on how you want to attach your Linux machine to your network. Typically, you'll keep DHCP selected and just move on to the next screen.

Even if you're using DHCP, you may want to define the hostname for your computer. If you don't, your computer will be known as 'localhost'.

7. Select firewall

Red Hat provides a software firewall as part of the regular installation process. Again, your network configuration will determine which setting you choose.

8. The next screens are again rudimentary – selecting the language support and the time zone.

Not much to say here.

9. Configure Root

The Linux version of Windows' Administrator account is called “root” (which is sometimes confusing, because there is also a directory called 'root'.) Root is more powerful, however, than a Windows administrator, which has two ramifications. First, it's critical that you assign a very strong password to root – someone who gains access to your computer as root can do anything. Second, don't use the root account unless it's absolutely critical – when you're doing system maintenance and/or administrator that can't be done any other way. Create a second account for your day to day use (as shown in Step 13).

10. Configure packages

Back in Step 3 I told you that you'd have the chance to select which software packages you were going to install. Here is that moment. The Package Group Selection dialog shows you which packages have been automatically selected based on which Installation Type you chose, and then to make modifications to that list. This works much like Windows' Add/Remove Software function.

11. Install

The last step is to click the “Go” button (it's probably called “Next” or something), start the install process, and watch the thermometer bars start scrolling across the screen. You'll be prompted to swap CDs a couple of times, but there aren't any restrictive licenses to peer through or activation routines to pray work.

12. Final steps – boot disk, video configuration

After all of your packages have been installed, you'll be prompted to create a boot disk (do it!) and confirm/select a video card. The first time you start your Linux system, you'll be prompted to run the Setup Agent. It will help you configure your system past a basic install.

13. Setup Agent

The Setup Agent allows you to create additional users, set the date and time, register with the Red Hat Network in order to get automatic patches, and install files from additional CDs (such as source code and documentation). That's it. You're now ready to use your Linux computer.

Installing Visual FoxPro on your Linux Workstation

It seems unusual to be able to run Visual FoxPro, a Microsoft product that has only run on Windows since the mid-1990s, on another operating system, but it can be done rather well. Here's how to do it.

1. How to run Windows Applications on Linux: WINE

Visual FoxPro only knows how to talk to Windows, right? Not anymore. Enter WINE, an open source software application that acts as an intermediary between Windows programs and the Linux operating system.

WINE intercepts calls to the operating system from Windows programs, translates those calls into their Linux counterparts, and sends the translated calls to Linux.

Linux receives those function calls, executes them, and returns the appropriate value or operation back to WINE. WINE then translates that result into its Windows counterpart, and sends the translated result to the Windows application.

The Windows application receives the translated result and goes on its merry way, never knowing that it's sitting on top of Linux, not Windows.

2. How to set up Visual FoxPro with WINE

In order to get Visual FoxPro to run on Linux, you use WINE, as described in the previous section. First, install WINE on your Linux machine. Second, install a newly licensed copy of Visual FoxPro on a Windows machine. (Note that you need a separate license for the copy of VFP that you're going to be installing on the Linux machine. You can't take the same copy you've installed on a Windows machine and use it for your Linux install.)

Third, copy some specific Visual FoxPro files from that Windows installation over to the Linux machine. Fourth, uninstall the files from the Windows machine. Fifth, Configure WINE to recognize VFP. And that's it!

Yes, it seems like there's a bit of extra work, but that's because WINE is still in alpha release. Over the next couple of years, as WINE matures, these steps will likely become automated and less cumbersome.

Now, let's look at each of these steps individually.

3. Uninstalling an old version of WINE

You can skip this section the first time you're installing WINE, since you don't have WINE on your system yet. However, since WINE is being upgraded regularly (about once a month), you'll want to keep current. In order to upgrade WINE on a system, you actually need to uninstall it and rebuild it from scratch.

The first thing you'll do is investigate whether there are any WINE RPMs on the system. Open a shell (terminal window) and make that session 'root' by issuing the 'su' command:

```
su
```

You'll be prompted for the root password. After entering it, that shell has root privileges, so be careful about what you type! Next, look for RPMs associated with WINE:

```
rpm -q --all | grep -i "wine"
```

If results are returned, for each package listed, issue the command

```
rpm -e --allmatches --nodeps <package name>
```

where <package name> is entered exactly (including upper/lower case) as it was in the first rpm command.

Next, regardless of whether there were any RPM results, it's time to manually remove anything else that was left laying around. Issue the commands

```
cd /
find -name wine
```

and wait until you get another prompt back to make sure the command has finished executing (it might take 2 or 3 minutes, depending on your system.)

Delete all files and directories. You can delete a file named "wine" (no extension) that's in the /usr/local/bin directory via this command:

```
rm /usr/local/bin/wine
```

You'll also find directories that contain lots of files. If you attempt to simply 'rm' those directories, you'll find yourself answering a "Are you sure?" prompt over and over again. In order to delete the directory called "wine" in the /home/yourname directory - and it's contents - without that incessant prompting, use the command:

```
rm -r --force /home/yourname/wine
```

The '-r' flag causes 'rm' to recurse through the directories, and the '--force' flag causes 'rm' to withhold the prompting.

There should now not be any remnants of WINE left on your system.

4. Installing WINE

Installing WINE requires four steps. The first is to download WINE to your system. The second is to install a patch that will take care of a specific VFP behavior for which a fix has not yet been included in the WINE source code. The third is to build WINE from source code. And the fourth is to configure WINE once it's been built.

First, prepare a location on your computer to hold the files you're going to download. Some people create a directory called "wine" underneath their home directory (for example, /home/yourname/wine) but I create a subdir under my home directory for my programs, like so: /home/whil/progs. So I created a directory for WINE under progs: /home/whil/progs/wine.

You can find the latest version of WINE at www.winehq.com. Click on "Source Code" under the "Download" link on the left side of the main page. You'll be greeted with a download page; click on the WINE icon or the Download Now link underneath it. You'll be sent to the ibiblio download site. Look for the latest build of WINE. As of this writing, the latest is 20031016. The file you're looking for will be named something like 20031016.tar.gz.

Download this file into /home/yourname/wine (or /home/yourname/progs/wine if you're following my preference - but from now on, I'll assume you just created /home/yourname/wine). This is simply a ZIP file but with a different extension. You can unzip it by typing

cd /home/yourname/wine

tar -xzfv Wine-20031016.tar.gz

The entire directory structure of Wine, including source code, documentation, and scripts that perform various tasks, is contained in a directory named /home/yourname/wine/Wine-20031016.

The next step is to apply a patch to the source code file before compiling the source code into the WINE executable. This patch fixes a problem with WAIT WINDOWs and tool tips in Visual FoxPro. Download the patch from Paul McNett's website at

http://www.paulmcnett.com/vfp/wine

and place it into the wine directory (next to the tar.gz WINE file.) In order to patch the source code, run the command (it's a 'zero' behind the letter 'p'):

```
cat vfpwinepatchwinsize | patch -p0
```

A number of text prompts will scroll by and then you'll be prompted for the name of the file to patch. Enter

```
wine/dlls/x11drv/winpos.c
```

and press Enter. The patch process will run and now you're done with this step.

The next step is to compile the source code into executables that can be run. Unlike VFP, where the build process is a matter of clicking a button after setting a property or three, building executables in C involves a number of processes. Fortunately, the WINE folks have put together a tool that will handle nearly all of the steps for you. Run this script by entering

cd wine

./tools/wineinstall

in your terminal window. A number of lines of information will scroll on by, and then you'll be prompted whether you want to build and install WINE as root. Answer "yes" (type the whole word in) and press Enter. The build process will then begin. This will take anywhere from 15 to 25 minutes, depending on your machine. Eventually you'll be prompted for the root password so that the install tool can put the WINE executables into other directories (/usr/local/bin, for example). Enter the root password, wait a few more minutes, and you'll be asked a couple of questions to complete the process. The answers you should provide are displayed in bold below:

```
Create local config file ~/.wine/config? (yes/no)
yes
Windows was not found on your system, so I assume you want a Wine-only
installation. Am I correct? (yes/no)
yes
Where would you like your fake C drive to be placed? (default is
/home/yourname/c)
/home/yourname/c
```

A few more prompts will scroll by, and eventually you'll get a final message

```
Installation complete for now.
```

At this point, Wine has been installed but it doesn't know anything about Visual FoxPro yet. Let's look at what has been done to your system.

First, you'll have the Wine source code and Wine executables in various directories. However, you won't have to worry about them directly. Next, you have a "C drive" directory structure (underneath /home/yourname, unless you put it somewhere else) that emulates the C drive that Windows would expect to find. For instance, you'll find a Program Files directory and a Windows directory. You'll soon put a VFP directory in this structure as well.

You'll also have a Wine configuration directory, probably named /home/yourname/.wine unless you put Wine somewhere else. This directory contains your Wine configuration file and your Windows "registry" - but, unlike Windows, it consists of a number of text files, so you can edit these easily and safely.

What's pretty interesting is that many of the files under the Windows directory on your fake C drive are actually just shortcuts (Linux calls them "symbolic links") to corresponding files in /usr/local that were placed there near the end of the Wine installation.

The final step is to grab some system files from Windows that VFP requires in certain instances. This is a stopgap measure while the Wine folks are working on replacing them with native Wine functionality. Where do you get them? You can't simply copy them from another machine that's running Windows - that's potentially a license violation. Instead, get the Windows installation disks that came with the computer you're running Linux on. (How do I know that the machine you're running Linux on came with Windows? When was the last time you were able to purchase a computer without Windows? Exactly my point.)

Create a directory named /home/yourname/wine/nativeDLLs directory, and put the files oleaut32.dll, msvcrt.dll, and mscomctl32.ocx into that directory. Then, in wine/c/windows/system, create a symbolic link to each of those files:

```
mkdir /home/yourname/wine/nativeDLLs
cd /home/yourname/wine/c/windows/system
ln -s ../../../nativeDLLs/oleaut32.dll   oleaut32.dll
ln -s ../../../nativeDLLs/msvcrt.dll   msvcrt.dll
ln -s ../../../nativeDLLs/mscomctl32.dll   mscomctl32.dll
```

Register mscomctl32.ocx using Wine's version of regsvr:

```
cd /home/yourname/wine/c/windows/system
wine regsvr32 mscomctl32.ocx
```

And we are now ready to install Visual FoxPro.

5. Installing VFP

The Visual FoxPro Installer won't run natively on Linux (big surprise!) - at least, not yet. (The VFP 5.0 installer recently started working, so it's just a matter of time.) As a result, you'll have to get your VFP files over to your Linux machine some other way. I simply installed VFP on a Windows machine, copied the entire directory structure to my Linux machine, under /home/whil/c/vfp8, and then uninstalled the original version.

Some people prefer to copy VFP to the Program Files directory under the fake C drive - if you do so, you'll need to preface spaces in directory names with a backslash, like so:

```
<some command> /home/yourname/c/Program\ Files/Microsoft\ Visual\
FoxPro\ 8
```

Next, you'll have to copy the VFP system files to /home/yourname/c/windows/system. These files are

```
msvcr70.dll
vfp8enu.dll
gdiplus.dll
vfp8renu.dll
vfp8r.dll
```

And that's it.

6. Configuring WINE for VFP

Now configure Wine to recognize Visual FoxPro. There are three pieces to this.

First, you'll need to change one parameter in the wine/config file. In the [wine] section, make sure there is the line

```
"ShowDirSymLinks" = "1"
```

The original config file has this line commented out (the line starts with a semi-colon).

The second thing to do is to add two lines that tells Wine about Visual FoxPro. Specifically, to tell Wine to make VFP think it's running on NT 4.0, and to use the Windows version of oleaut32.dll instead of its own built-in version.

Open up the Wine config file, /home/yourname/wine/config (there's no extension on the file named 'config'). You can do this with your favorite Linux editor - mine is gedit, found under "Accessories | Text Editor".

The lines you should make sure are at the end of the config file look like this:

```
[AppDefaults\\dcom95.exe\\DllOverrides]
"ole32" = "native"
; Visual FoxPro 8
[AppDefaults\\vfp8.exe\\Version]
"Windows" = "nt40"
[AppDefaults\\vfp8.exe\DllOverrides]
"oleaut32" = "native, builtin"
```

Now let's test it out!

7. Using VFP on Linux

In order to actually run Visual FoxPro, we're going to go through Wine. Issue the following commands (assuming that you installed the VFP 8 files in a directory named "vfp8"):

```
cd /home/yourname/c/vfp8
wine vfp8
```

If all goes well, you should see the Visual FoxPro IDE display in a few seconds. Now it's time to roll up your sleeves and go to work!

The first thing you'll want to do is clean up your environment a bit. You'll want to undock your windows. Right click on the Command Window's title bar and uncheck "Dockable". Next, Alt-Tab to a different window on the desktop and then back to VFP, and you'll see the cursor display in the Command Window.

Next, determine what the current directory is - type

```
? cd
```

into the Command Window and press Enter. You will see that you're in the vfp8 directory. All good FoxPro programmers know that it's a bad idea to do anything in the VFP directory itself, so you'll want to switch to another directory automatically when you start up VFP. Create a data directory under the fake C drive directory, and then create a config.fpw file in the vfp8 directory, and enter your startup commands in it as usual, including setting your default directory to your new data directory.

Now it's time to start working with VFP in earnest. Try creating a simple application, with a main program, a project file, some forms and a couple of local data tables.

At some point, you may need to deal with what VFP sees in terms of operating systems. Enter

```
? os()
```

in the Command Window. You should see "Windows NT 4.0" on the desktop. That's right - for all practical purposes, VFP thinks it's running on Windows. However, you may need to know if your application is truly running on a Windows box or if it's being faked out. Some of you may remember the _DOS, _MAC, and _UNIX system variables for FoxPro 2.x versions. Paul McNett has a clever trick for setting a _WINE variable. Put the following code in your startup routine:

```
public _WINE
_WINE = "wine" $ lower(getenv("_"))
```

Then _WINE will be set to .t. when running on Wine and .f. otherwise.

8. Deploying Applications

The preceeding discussion addresses the use of VFP in development mode. The way you deploy a Visual FoxPro application on Linux depends on a variety of factors.

Generally, Visual FoxPro applications are deployed by placing the EXE you create that contains your custom code together with the Visual FoxPro runtime files on a user's machine. The Visual FoxPro End User License Agreement (EULA) explicitly has provided for this method of deploying an application – without payment of additional runtime fees – since the days of FoxPro.

However, Microsoft has become threatened by the increasingly popularity of Linux, and has begun to modify its EULAs in order to restrict the distribution of certain files to Windows environments. The language in the EULA changed from VFP 6 to VFP 7, and then again to VFP 8, including the restriction for the first time between 6 and 7, and then tightening the restriction from 7 to 8. However, the EULA's wording is vague and ambiguous, and is open to interpretation a number of ways. Requests to Microsoft for clarification have been met with additional vague and ambiguous statements, and they have refused to answer additional requests for clarification, insisting that the customer consult an attorney.

As a result, deployment techniques depend on which version of Visual FoxPro you're using. If you're using VFP 6 or earlier, you can deploy simply by including your EXE and the runtime files, and running them on a Linux box via WINE, just as with the development version. The EULA for VFP 6 and earlier does not attempt to restrict this activity at all.

If you're running VFP 7 or 8, however, there is some question as to whether deployment of the runtime files is allowed. The safe method, then, would be to develop in VFP 7 or VFP 8, taking advantage of the IDE's productivity features, but not the language enhancements past VFP 6. Then, in order to deploy, use VFP 6 to build the EXE and deploy it along with the VFP 6 runtime files.

Alternatively, if you needed functions only available in VFP 7 or 8, you could buy a license for Visual FoxPro for each machine you're going to deploy on, and run the application from within the IDE.

* EOF