

Chapter 2

How Windows Runs FoxPro

I showed you how Windows is split into two codebases—the Windows 95 Kernel and the Windows NT kernel. Now look at how those kernels provide support for legacy applications like FoxPro—and some of the problems that can occur.

As you've seen, modern versions of Windows provide support for MS-DOS and Win16 applications by the use of Virtual Machines. It's useful to understand general Windows architecture and how VMs fit into it to help fix problems.

An overview of how MS-DOS and Win16 applications run in the Windows 95 kernel

Figure 1 shows in a simplified form how the Windows 95 kernel operating systems (that's Windows 95, Windows 98, and Windows Millennium) are organized. Take a look some of the relevant components from a FoxPro perspective.

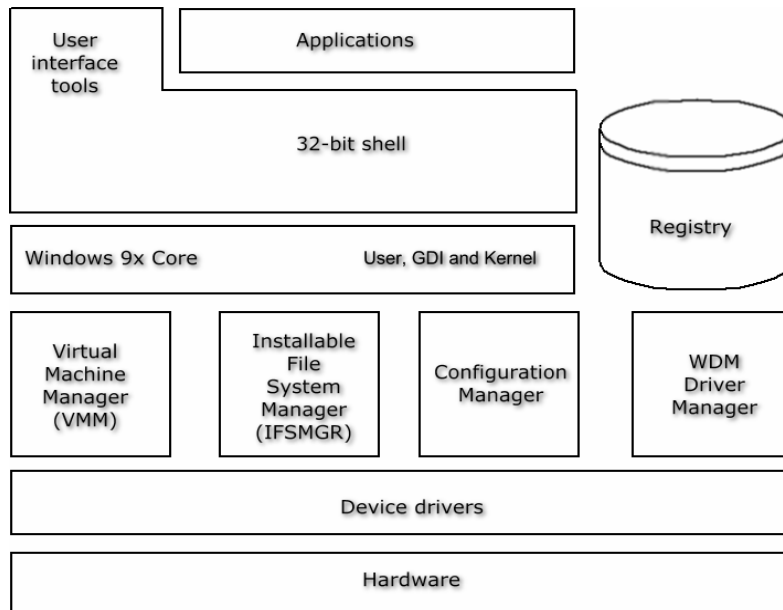


Figure 1. Components of the Windows 95 architecture.

At the top level are applications. Generally, you see those applications via the Windows GUI (Graphical User Interface), also called the 32-bit Shell. The second layer in the diagram represents the Shell.

Windows core components

Below the User Interface are the Windows 98 core components. There are three of these—User, Graphical Device Interface (GDI), and Kernel.

User

The User component handles all inwards and outwards communication via the mouse, keyboards, serial, parallel, and USB ports. It also handles output to the GUI at a high level, controlling icon and Window placement. A lot of the User component remains written in 16-bit code for backward compatibility. This is why it is not entirely accurate to refer to the Windows 95 kernel operating systems as being ‘32-bit’.



You will sometimes see Windows errors occurring ‘in module USER’. When this happens, it is a good idea to troubleshoot the areas the User component is responsible for. For example, are the mouse and keyboard drivers correct?

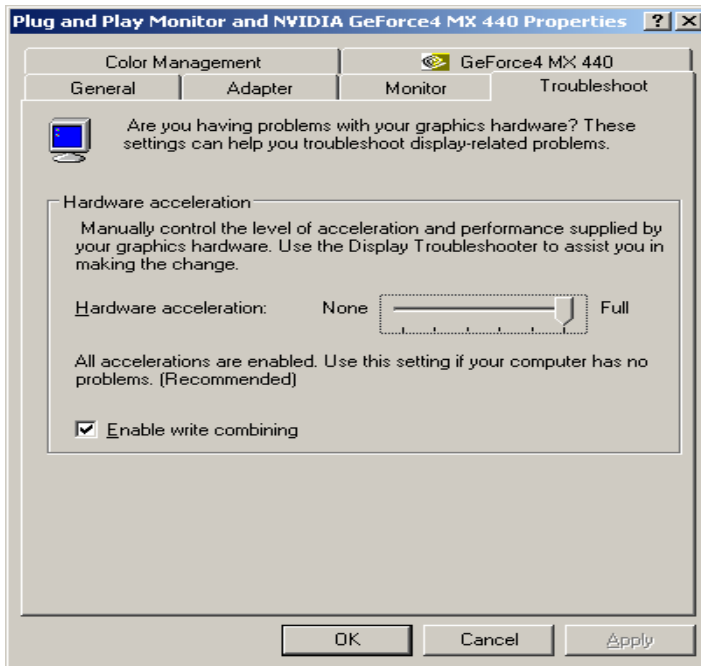


Figure 2. *Adjusting the graphic acceleration settings.*

Graphical Device Interface (GDI)

GDI handles all bitmapped output, mainly to the screen but also to devices such as printers. It provides standard libraries both Windows and applications can use. Examples include drawing

graphics primitives like curves, lines, and fonts. Most of the code in the GDI is 32-bit for optimal performance.

Windows protection errors in the GDI component sometimes occur. Also, users can encounter display-related issues within FoxPro for Windows applications, with errors such as 'Position Is Off The Screen'. Areas for investigation would include graphics card drivers, printer drivers, and corrupt or missing font files. You can also lower the graphic acceleration settings in Windows to address these problems, under Control Panel / System / Performance, as in **Figure 2**:

If the error happens when your FoxPro application is doing something printer-related, then the problem is almost certainly the printer driver. Try using a basic Microsoft driver for the relevant family of printers, found on the Windows installation CD-ROM. For example, various releases of PCL6 series LaserJet drivers supplied by Hewlett-Packard caused severe problems due to a memory leak. They can cause havoc in some FoxPro for Windows applications just by being installed on a workstation, regardless of whether that application uses them or not. A workaround is to use the standard LaserJet 4 or LaserJet III driver supplied on the Windows CD-ROM.

Kernel

The Kernel is the real engine of operating systems. It handles vital tasks like memory management and file I/O. It also loads and executes EXE and DLL files.

Memory is addressed under the Windows 95 kernel by using 32-bit numbers. To provide support for older 16-bit applications like FoxPro, the Kernel also translates 16-bit memory addresses into 32-bit equivalents using a process called 'thunking'.

Repeated Windows errors in the Kernel module generally indicate something fairly serious is wrong. This could include faulty hardware such as memory. It could also be due to a corrupt FoxPro application executable program file. If the problem is happening in a FoxPro application, try to isolate the area where the problem is occurring. Start Windows in Safe Mode to see if another installed program is causing a conflict. Safe Mode loads only core Windows services with a basic low resolution desktop. Try running the FoxPro application on its own and if the problem doesn't occur, at least you know it's not intrinsic to the application. Another option is to run the MSCONFIG utility (Start | Run | msconfig) and stop unnecessary items from loading via the 'Startup' tab, as in **Figure 3**.

Other Components

Of the other components involved, the most important by far is the Virtual Machine Manager (VMM). It essentially allows us to run our old applications on Windows.

Virtual Machine Manager

The VMM provides each application with the resources it needs to run, and keeps those resources isolated from those used by other running programs. This component provides multitasking. Each application is presented with a Virtual Machine (VM) that appears as a separate PC complete with its own resources to the 16-bit application. The Windows 95 kernel operating systems have a main VM, called the System VM, where all system processes run. Each 32-bit application runs in its own separate VM. 16-bit Windows executables (like FoxPro for Windows) all run in the Win16 VM and share the same memory. MS-DOS applications, like FoxPro for DOS, each get their own VM.

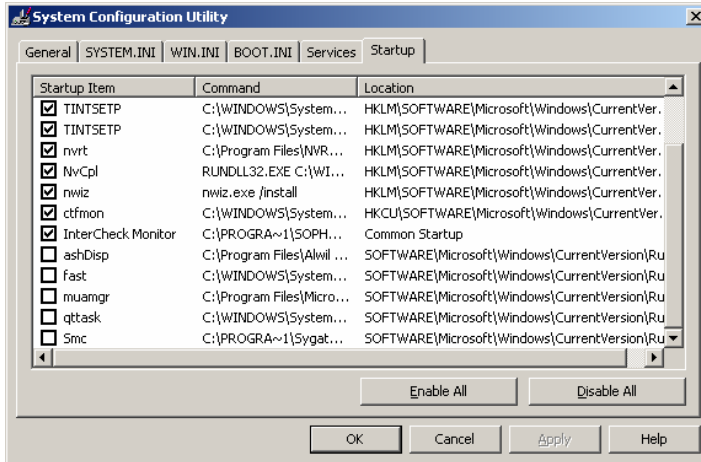


Figure 3 Using the MSCONFIG utility to stop startup items from loading.

If an application or driver tries to access memory reserved for another process, or attempts to execute an illegal instruction, the Intel 80386 and later CPU's will detect this and throw a General Protection Fault. These are picked up by Windows and displayed on the so-called 'Blue Screen Of Death.'

FoxPro is a high-level language and programs developed using it cannot access memory locations directly. As a result, Protection Faults and Fatal Exception errors that occur when a FoxPro application is running are usually due to external factors. Badly written printer drivers are one major cause. However, it is also possible damaged FoxPro CDX index files might be the problem if there is corruption of an internal pointer causing FoxPro to try to access an illegal memory location. In that case, removing all indexes (DELETE TAG ALL in FoxPro) and recreating them will solve the problem. Most FoxPro applications should have a re-indexing or index destruction and recreation function included if a copy of FoxPro itself is not available.

When the problem involves the VMM, you may see messages mentioning a "Protection Fault in VXD VMM..." Another very common example under Windows 98 is "Fatal Exception 0D in module VMM..." or "Fatal Exception 0E in module VMM..." They are mostly caused by problems with drivers for certain installed hardware. Microsoft Knowledge Base article Q150314, titled "What Are Fatal Exception Errors," provides a good starting point for further investigation. Microsoft Knowledge Base article Q82710 has more advice on investigating General Protection Faults.



A good way of getting Microsoft Knowledge Base articles is to send an empty e-mail to 'mshelp@microsoft.com', with the article number as the subject line. The article will be automatically e-mailed back to you. Alternatively, search on the article number using the indispensable Google search engine at <http://www.google.com>.

An overview of how MS-DOS and Win16 applications run in the Windows NT kernel

MS-DOS has huge limitations on modern hardware. By extension, the Windows 95 kernel operating systems built upon MS-DOS do also. Microsoft had a clear requirement to produce a modern, secure operating system aimed at business while at the same time not abandoning customer investment in MS-DOS and Win16 applications. The answer was to split Windows into two streams, the by now familiar Windows 95 kernel and Windows NT kernel. The former emphasised compatibility and ease-of-use, the latter resilience and security.


 *People often say they want reliable systems. They don't. If their system crashes every 10 minutes, then they'll be able to set their watches by it reliably. What they really want are resilient systems.*

Figure 4 shows in a simplified form how the Windows NT kernel operating systems (that's Windows NT, Windows 2000, and Windows XP) are organized.

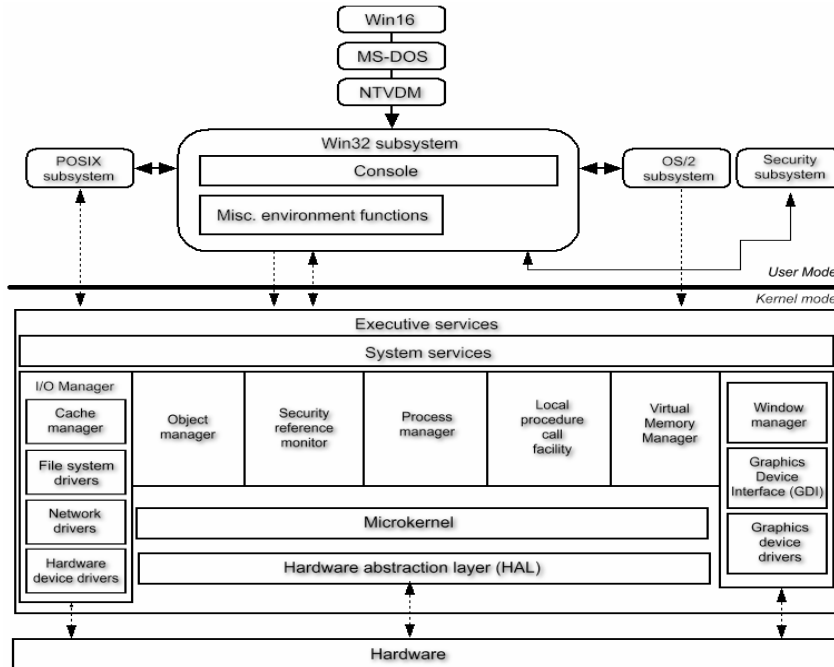


Figure 4. The Windows NT Executive Architecture.

The NT Kernel way of doing things uses a layered approach called the NT Executive. Hardware and memory used by system processes are isolated from applications. In Figure 4, everything below the heavy black line is isolated from everything above.

Access to hardware, say by a printer driver, can occur only through the Hardware Abstraction Layer (HAL). Applications cannot access hardware directly, and attempts to do so are trapped by Windows.

To provide more stability, Windows NT implements Kernel Mode and User Mode.

Kernel Mode

Applications running under this highly privileged mode have direct access to all memory and hardware. It is reserved for the components of the operating system, and is also known as Supervisor Mode, Protected Mode, or Ring 0. This mode is provided at the CPU level on Intel 80386 and later processors. Ordinary applications or drivers that attempt to access this mode will cause a Protection Fault.

User Mode

Applications run in User Mode, and are only able to access memory in the address space provided to them by the operating system. The NTVDM subsystem (described below), which FoxPro applications run under, operates in User Mode.



Lack of direct access to hardware under the Windows NT kernel can cause problems for some FoxPro for DOS and FoxPro for Windows applications, usually those that attempt access to the serial ports for label printers, cash tills, and so on. They may well address the ports using third-party FLL libraries written in the C language—SilverFox is a well-known example. If this is the case, it may be easier to run the application on a Windows 95 kernel operating system. An alternative is to port the application to Visual FoxPro, which can use MSCOMM32, the standard Windows communication library.

The MS-DOS NTVDM

Under the NT kernel, an MS-DOS application runs in a Virtual DOS Machine process called an 'NTVDM'. This emulates an Intel 80486-based PC, and each MS-DOS application runs in its own NTVDM process, using its own memory space. Intel 80486 CPUs actually provide a special mode called Virtual86 mode that allows most instructions in an MS-DOS application to execute directly, i.e. without any need for an additional layer of emulation that would slow everything down.

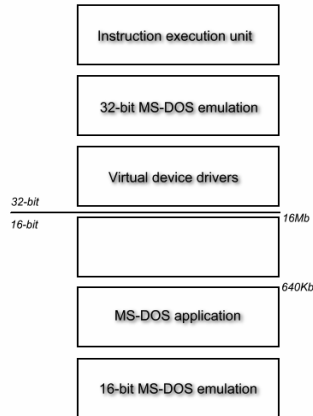


Figure 5. The structure of the MS-DOS NTVDM.

As **Figure 5** shows, the NTVDM emulates the 640Kb base memory system so beloved of old MS-DOS salts. Also worth noting is the 16Mb upper limit on addressable memory.

Each NTVDM running an MS-DOS application has three threads: An Application Thread under which application instructions are executed, a Heartbeat Thread that simulates timer interrupts to the MS-DOS application, and a Console Thread that handles console (display) I/O for the application.

You can see the NTVDMs easily by pressing CTRL-ALT-DELETE in Windows NT/2000/XP and selecting ‘Processes’ in the Task Manager, as in **Figure 6**. If you were experiencing a problem where the your application was starting but freezing straight away, this is where you would look to check on what was happening with NTVDM, for example with processor utilization.

WOW–Win16 on Win32

Win16 applications like FoxPro for Windows run in NTVDMs, which are slightly different than their MS-DOS counterparts (see **Figure 7**).

All Win16 applications run as separate threads inside one Win16 NTVDM. They all share the same memory space. The Win16 NTVDM contains two system threads: the WOWEXEC thread, which starts the individual application threads, and a Heartbeat Thread, which supplies timing interrupts.

WOWEXEC is also easily seen running in the Task Manager, as in **Figure 8**.

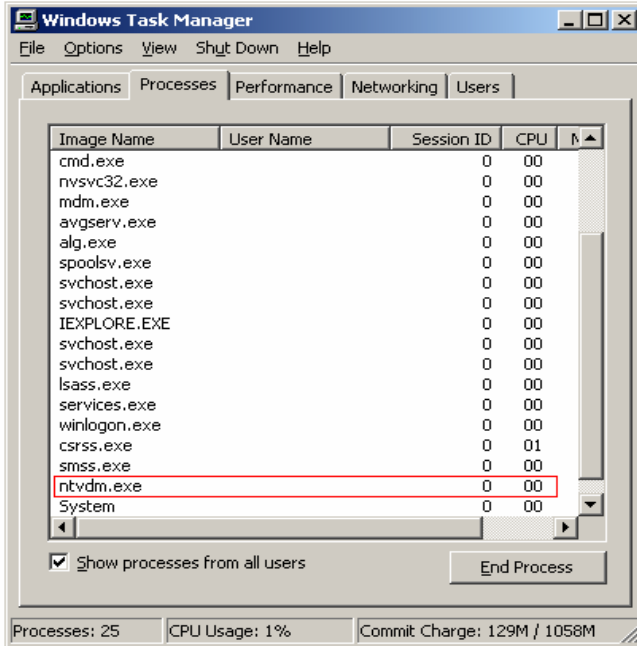


Figure 6. The NTVDM process running in Task Manager.

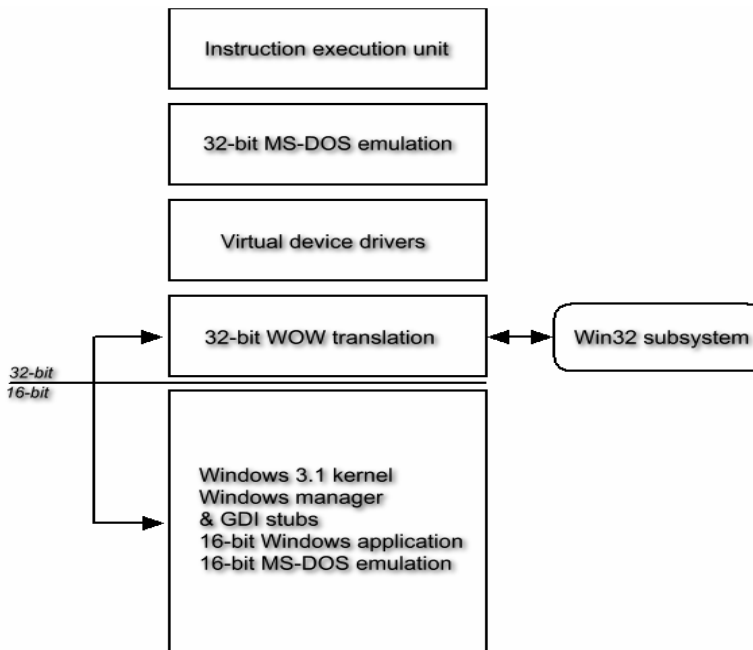


Figure 7. The structure of the Win16 NTVDM.

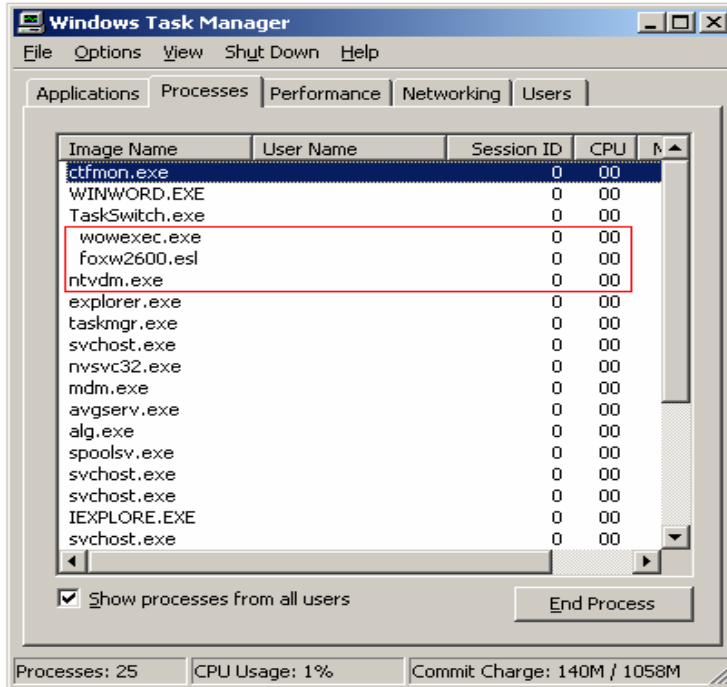


Figure 8. The WOWEXEC process running in Task manager.

In Figure 8 the indented WOWEXEC.EXE and FOXW2600.ESL items belong to the NTVDM process below them (the order of display is unimportant). FOXW2600.ESL is the FoxPro for Windows runtime library and its presence indicates a FoxPro for Windows application is running. Apart from the academic exercise of being able to see what's running under NTVDM, the Task Manager is useful in determining if a FoxPro application started at all, or which one has gone to 100% CPU utilisation. The 'End Process' button is also useful for terminating crashed FoxPro applications.

It is possible to force a Win16 application to use its own memory space, effectively making each legacy application run in a separate NTVDM process (remember Win16 applications use a shared memory space by default). This could help resolve problems where one FoxPro for Windows application is causing a memory problem like a Windows Protection error and you need to isolate which one it is. This is done by creating a shortcut to the Win16 executable file. Then, under the shortcut's properties, click on the advanced button and select the 'Run In Separate Memory Space' checkbox. See **Figure 9** for an example from Windows XP.

A common problem with FoxPro for DOS applications running under the Windows NT Kernel is very high CPU utilization. The CPU column as seen in Figure 6 might display up to 99%. This is generally due to the internal FoxPro clock, usually visible in the bottom right corner of the application. It can sometimes be turned off by looking for (or creating) a text file called CONFIG.FP in the same folder as the application, opening the file with a text editor like Notepad, and entering this line:

```
CLOCK = OFF
```

This may not work if the application specifically sets the clock on internally. There is at least one commercial utility available that addresses this problem, namely TameDos found at <http://www.tamedos.com>. There's more about CONFIG.FP later.

On some machines the NTVDM process will stall completely when a FoxPro for Windows application is trying to start. The NTVDM process will be at 100% CPU utilization. You will see in the next chapter how to edit CONFIG.NT to overcome this.

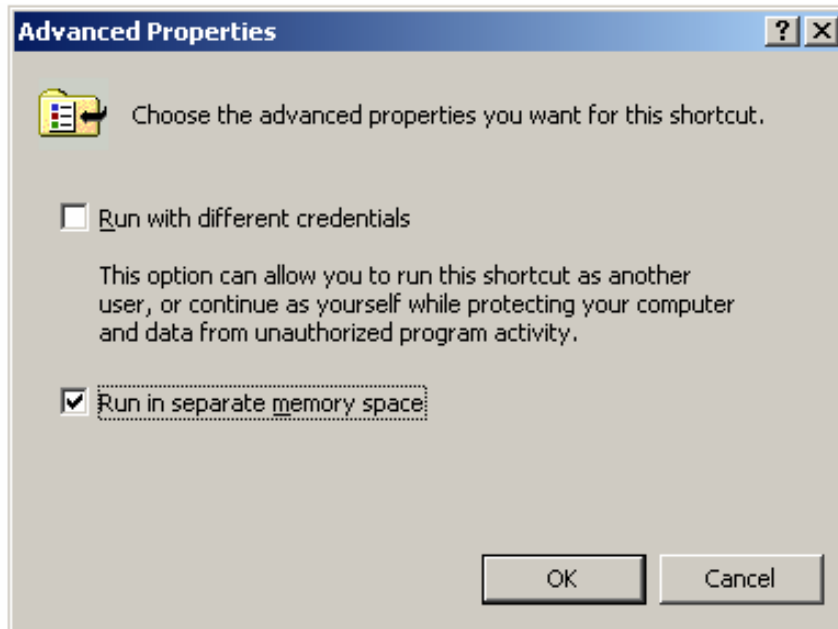


Figure 9. *Running a Win16 application in a separate memory space.*

Conclusion

Now you are familiar with the general principles of how FoxPro applications run in Windows. In the next chapter you will see how to configure both FoxPro and Windows so they get along.