

NT stands for New Technology. It could just as easily stand for Networking Technology, because Windows NT was designed from the ground up to be used on a network. NT was released in 1993, and was aimed at the corporate market where Windows 3.1 had failed to make significant inroads.

There are two flavors of NT, one for the server and another for the workstations. The A+ exam does not cover NT Server, so we will only mention it in passing and concentrate on the NT Workstation version and its successor, Windows 2000. Throughout this chapter, NT will mean NT Workstation Version 4 unless it specifically says NT Server.

What would have been NT version 5 became Windows 2000 (covered in the next chapter), so the final version of NT was Version 4. There have also been several upgrades to that version, called **Service Packs**.

The first thing you are likely to notice about a booted-up NT system is that it looks virtually identical to Windows 9x. That's because NT 4.0 borrowed the Windows 95 GUI, to the relief of everyone concerned. As far as the interface goes, it will be easier to just talk about the slight differences than to repeat everything we said in the last three chapters. Anything that isn't covered regarding the user interface, you can assume will look and work like it did in Windows 9x.

Beneath the surface though, there are some profound differences between 9x and NT in the areas of boot-up, security, file structure and disk management. We will soon enough be into each of these topics in agonizing detail.

FUNCTIONAL DIFFERENCES

One of the basic differences with NT is that it can work with multiple processors. We can take this to mean two things, both of which are true. First, NT is not restricted to systems with Intel-type CPUs, but can also work on **RISC** systems.

RISC stands for **R**educed **I**nstruction **S**et **C**omputer. The idea is that the CPU will be faster by using just a few simple instructions, and the software can put these together in combinations to equal the more complex instruction set of say, a Pentium CPU. Some examples of RISC CPUs are the **Alpha** and **MIPS** processors.

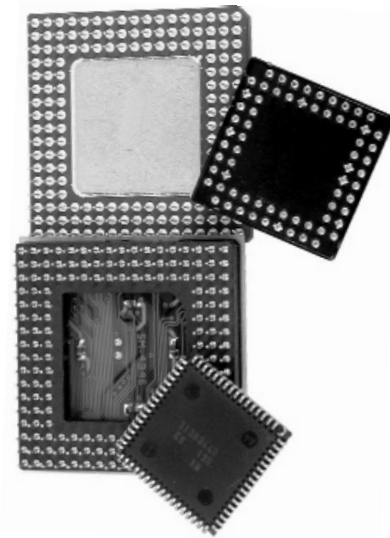
The second meaning of 'multiple processor' is that there is more than one CPU in the system. NT Workstation can be used in systems with two processors, while NT Server and Windows 2000 can handle up to 32 processors.

NT can run on any of the IBM-compatible PC systems including PS/2, but not on the original XT systems. Heck, they had to draw the line somewhere.

MULTIPROCESSING

For multiple CPUs, NT uses what is called **Symmetric Multiprocessing** or **SMP**. In this mode, the operating system can run on either CPU, and tasks and users can be divided between CPUs in the way that the OS finds most efficient.

The other mode, not used by NT, is called **Asymmetric Multiprocessing** or **ASMP**. Here, the operating system runs on one CPU and applications are assigned to the other. While this method is initially easier to design, it is less efficient and not as flexible to move to a variety of platforms.



MULTITASKING & MULTITHREADING

Some people use the term 'multiprocessing' when they really mean multitasking. NT uses pre-emptive multitasking similar to Windows 9x, but it is also capable of **Multithreading**. A **Thread** is a single action being done by the processor. Each application that is opened creates at least one thread, but possibly more. For example, one thread is used to maintain a window for the application. Another thread may be used for an action taking place in that window, while the application is printing a job in the background which takes yet another

.....

thread. Windows NT has the ability to juggle all of these threads simultaneously, allowing applications to make more efficient use of their share of CPU time. Multi-threading has also been included in Windows 98.

OS SUPPORT

NT is also able to run applications that were written for other operating systems. This includes not only Windows 9x but also Unix, DOS and 16-bit Windows versions as well as 16-bit programs written for IBM's OS/2 operating system.

The reason it can do this is because of the layered way NT is structured. At the heart of it is the **NT kernel**, which handles the basic file I/O, task management and virtual memory (swap file) duties. Between the kernel and the applications are a set of three **Subsystems**, which translate between the kernel and the applications. Another name for one of these subsystems is **API**, or **Application Program Interface**.

The main one is the WIN32 API that is used for applications designed specifically for NT or other 32-bit Windows versions.

An **OS/2** subsystem can run applications from OS/2 1.x, but not those for OS/2 version 2 or higher. The **POSIX** subsystem runs applications designed for Unix.

DOS and 16-bit Windows programs use WIN32, but they run inside a special version of a virtual machine called a **VDM** or **Virtual DOS Machine**. This is also called a **DOS Emulator**, because it emulates, or gives the appearance of, a DOS system by setting up a 1MB block of memory and making the application think that's all there is.

One nice thing about running these 16-bit programs in a VDM instead of conventional memory is that there isn't any more conventional memory. Or extended, expanded, upper or high memory. It's just memory, from start to finish. Hooray!

HARDWARE ABSTRACTION LAYER

The other layer in this kernel sandwich is between the NT kernel and the hardware. This is called the **HAL**, or **Hardware Abstraction Layer**. I presume there is no relationship to the computer HAL in the movie 2001, A Space Odyssey, the one that went crazy and tried to kill Dave the astronaut.

This HAL is a software interface between the NT kernel and the hardware of the system, and there are different versions for different CPUs. The HAL is the reason why NT can be used on non-Intel CPUs, because it will have a different HAL for the Alpha, MIPS or other CPU types.

You should be aware that the HAL can cause problems for certain applications, especially older 16-bit ones, that were written on the assumption that the application would be able to directly manipulate some piece of hardware. This is especially true of

.....

many DOS-based games, which assume they are the only program running and try to hijack the video and sound to tweak performance.

VXD

In NT, nothing and nobody talks to the hardware without going through the HAL and the NT kernel. To avoid shutting down any application trying to access the hardware, NT supplies something called a **VxD** or **Virtual Device Driver**. This is a piece of software that simulates the hardware being accessed so that the program will think it has adjusted the hardware and will go on its merry way. NT includes VxDs for the basic peripherals such as mouse, keyboard and video, but programs trying to access other peripherals may find themselves shut down.

THUNKING

One of the biggest hurdles in having a system support both 16-bit and 32-bit applications is that those applications might have to talk to each other. NT deals with this by adding a layer called the **Thunk Layer**. It's called that because translating between 16 and 32-bit data is called **Thunking**, and beyond that, don't ask me.

Anyway, the kernel and application interfaces have both 16 and 32-bit modules, and they interface with the Thunk layer so that the applications don't actually come face-to-face. It's a very NT way of doing things.