OPERATING SYSTEMS

ITE 208

Topic: WINDOWS PHONE KERNEL (WINDOWS NT)

SLOT: A1

SUBMITTED TO:

PROF. ARIVUSELVAN K. (OS MENTOR)

SUBMITTED BY:

MUKUL DEV

(13BIT0269)

**The architecture of Windows NT**, a line of operating systems produced and sold by Microsoft, is a layered design that consists of two main components, user mode and kernel mode. It is a preemptive, reentrant operating system, which has been designed to work with uniprocessor and symmetrical multi processor (SMP)-based computers. To process input/output (I/O) requests, they use packet-driven I/O, which utilizes I/O request packets (IRPs) and asynchronous I/O. Starting with Windows 2000, Microsoft began making 64-bit versions of Windows available—before this, these operating systems only existed in 32-bit versions.

Windows NT 5.0 is a massive release, with significant new operating system and network functionality being added to many areas of the system. It is not, however, a rewrite-many aspects of the kernel system architecture remain the same or are being extended. In other words, the fundamental operating system architecture is not changing in Windows NT 5.0. The article is intended as a brief introduction to the NT architecture, including those features-notably plug and play, the job object, and 64 bit large memory support-that necessitated extensions to the executive and the kernel. The author briefly describes major new features in NT 5.0, such as the Active Directory and the Distributed Security Services. The companion articles describe major new extensions to the NT file system and the Microsoft Clustering Service

**Kernel**

The kernels sits between the HAL and the Executive and provides multiprocessor synchronization, thread and interrupt scheduling and dispatching, and trap handling and exception dispatching; it is also responsible for initializing device drivers at boot up that are necessary to get the operating system up and running. That is, the kernel performs almost all the tasks of a traditional microkernel; the strict distinction between Executive and Kernel is the most prominent remnant of the original microkernel design, and historical design documentation consistently refers to the kernel component as "the microkernel".

The kernel often interfaces with the process manager. The level of abstraction is such that the kernel never calls into the process manager, only the other way around (save for a handful of corner cases, still never to the point of a functional dependence).

**Kernel mode**

Windows NT kernel mode has full access to the hardware and system resources of the computer and runs code in a protected memory area. It controls access to scheduling, thread prioritization, memory management and the interaction with hardware. The kernel mode stops user mode services and applications from accessing critical areas of the operating system that they should not have access to; user mode processes must ask the kernel mode to perform such operations on their behalf.

Kernel mode consists of executive services, which are made up of many modules that do specific tasks, kernel drivers, a kernel and a Hardware Abstraction Layer, or HAL.

**The NT Executive**

The NT Executive takes care of the important tasks that are vital to the entire system. This includes services such as object management, virtual memory management, I/O management, and process management.

**The Executive has the following components**

* **Kernel** - The kernel is ultimately responsible for all actions on the system and almost all functions on the system pass through the kernel. The kernel is responsible for scheduling all system actions; handling system interrupts and handles processor exceptions.

On a multiprocessor system, a copy of the kernel actually runs on each processor. Because the kernel is involved in almost every action taken on an NT system, critical portions of the kernel are written in assembly language.

* **Object Manager** - The Object Manager piece of the NT Executive is used to create, modify, and delete objects used by all the systems that make up the NT Executive. Objects are abstract data types that are used to represent operating system resources. Objects can be concrete, such as device port, or they can be more abstract, such as a thread. When an object is created, it is given a name by which other programs can access the object.
* **Process Manager** - The Process Manager is responsible for creating, removing, and modifying the states of all processes and threads. It also provides information on the status of processes and threads to the rest of the system. When an application is started, it is created as a process, which requires a call to the Process Manager. Because every process must have at least one thread, the Process Manager is invoked again to create a thread.
* **Virtual Memory Manager** - The Virtual Memory Manager (VMM) provides management of the system's virtual memory pool. Virtual memory is a scheme that allows disk resources to be used instead of physical system memory.
* **Local Procedure Call Facility** - LPC is a mechanism that enables two threads in different processes to exchange information.
* **Security Reference Monitor** -The Security Reference Monitor (SRM) is the bedrock of the all security on a Windows NT system and is responsible for enforcing all security policies on the local computer.
* **I/O Manager** - The I/O Manager is responsible for coordinating and processing all system input and output.

**Hardware Abstraction Layer**

The Hardware Abstraction Layer (HAL) is a software interface between the hardware and the rest of the operating system. The HAL is implemented as a dynamically-linked library (DLL) and is responsible for shielding the rest of NT from hardware specifics such as interrupt controllers and I/O interfaces. This abstraction makes NT more portable because the rest of the operating system does not care what physical platform it is running on. Each hardware platform that NT runs on requires a specialized HAL. The design intent is that when NT is ported to new processor architecture, the HAL gets rewritten for the new processor, but the rest of NT can simply be recompiled, thus making NT extremely portable.

Although the intent of the HAL is to reduce the amount of hardware dependencies and make NT more portable, in reality, it's not always quite so simple, but by minimizing the dependencies on physical hardware characteristics the designers of NT have reduced the time and effort needed to move the operating system to a new platform.

The HAL can only be accessed by components of the NT Executive, and is never called directly by user-mode programs. Also, the HAL is intended to be the only piece of software on an NT system that is permitted to talk directly to the hardware. The advantage is that rogue programs cannot purposefully or accidentally write information to the hardware and cause a system crash. Also, preventing programs from reading information directly from the hardware helps to support NT's security model.

Although the goal in Windows NT is to have all hardware-related calls go through the HAL, the reality is that a small number of device driver and kernel calls bypass the HAL and directly interact with the hardware.

The downside of the HAL model is that it is the biggest single cause of incompatibility with older DOS and Windows programs, which were in the habit of reading and writing directly to hardware. However, this incompatibility is a small price to pay for the protection and portability afforded by the HAL.

**Environment Subsystems**

The protected environment subsystems act as mediators between the user-level applications and the NT Executive.

The Environment Subsystems allows Windows NT to effectively act as if it were a different operating system. In Windows NT, there are three protected environment subsystems: Win32, POSIX and OS/2.

Although you might see the Win16 and DOS personalities included in a list of protected environment subsystems, they are actually both part of the Win32 subsystem.

Each environment subsystem keeps track of its own processes and works independently of the other subsystems. Each application can run only in the subsystem for which it was designed. When you launch an application in Windows NT, it examines the file and determines which subsystem to run the application in.

Win32 is the native and primary subsystem for Windows NT. The basis for this subsystem is the Win32 set of APIs, which were written during the development of the NT product. Many of these APIs are direct extensions of their Win16 counterparts.

The POSIX (portable operating system interface) was developed by the IEEE as a standard for use on UNIX systems.

There are many levels of POSIX compliance ranging from POSIX.0 to POSIX.12. These levels represent an evolving set of proposals, not all of which have been ratified as standards.

The POSIX subsystem in Windows NT is POSIX.1 compliant. POSIX.1 compliance requires a bare minimum of services, which are provided by Windows NT. When a POSIX application runs on Windows NT, the POSIX subsystem is loaded and it translates the C language API calls— for POSIX.1 support— Win32 API calls, which are then serviced by the Win32 subsystem.

Because of the limited nature of POSIX.1, the POSIX subsystem on Windows NT does not provide any support for networking or system security. Many people feel that the inclusion of the POSIX subsystem was really a marketing ploy to increase NT's market penetration.

For the first year and a half of its design, the OS/2 subsystem was scheduled to be the default and primary subsystem for Windows NT. However, when the decision was made to give NT the Windows interface and to build it as the successor for the Windows platform, the emphasis on OS/2 support was diminished.

The result was a second crippled subsystem. The OS/2 subsystem is not able to run OS/2 2.x graphical applications and the OS/2 subsystem only works on Intel-based systems, not on RISC platforms.

Conclusion

The modular design of the Windows NT architecture makes it both portable and scalable. Windows NT's hardware abstraction layer allows the operating system to run on different hardware platforms. Currently, Windows NT runs on Intel X86, Digital Equipment Corp.(DEC) Alpha, MIPS RISC (reduced instruction set computing), and the PowerPC series of microprocessors jointly manufactured by IBM Corp. and Motorola. Besides being highly portable, Windows NT supports scalability, which allows the operating system to effectively use multiple processors. Thus, when network managers evaluate Windows NT Server as a platform for different applications, it is important for them to note that they have several options for retaining their investment as applications grow.

For example, because of its scalability, network managers could replace a uniprocessor Intel Pentium motherboard with a dual- or quad-processor motherboard. If this replacement does not provide the necessary level of processing power, network managers might consider migrating hardware to a high-level PowerPC or a DEC Alpha based computer. If that migration is required and the applications continue to grow, network managers could use multiple processors to ensure scalability.