Microsoft Operating Systems Management Correlation

An OS to Hardware Interactions Overview

By

Kenneth G. Foster

May 23, 2010

Abstract

This paper examines Microsoft Windows in the context of the operating system and how it interrelates to the supported hardware. An overview pertaining to memory management, processor scheduling, input / output, interrupts, system integrity, the file system implementation and security are examined and clarified.

This paper examines the Microsoft Windows operating system in the context of its direct interactions and management of the support environment. The operating system is the broker for activities on the hardware. Microsoft has done an excellent job insulating the user from the highly complex transactions that occur while using the system. In this paper we will address the mechanisms involved in how the processor, memory, and Input / Out interact. We will examine the symbiotic relationship between process manage and control flow. Finally we will discuss availability and sustainability of both data and hardware.

An operating system is the program that controls how the various physical hardware components of a system interact with instructions, referred to as software, execute. This is accomplished through a complex system of interconnected management algorithms which work in concert to execute and enforce policies. These policies control the security of the system, what instructions are executed in which order on the processor, the management of memory, and the communication between Input / Output (I/O) devices. This orchestra of management is designed to ensure data integrity, system availability, and the most efficient method of instruction execution (Stallings, 2009 p. 109). Without these efficiencies, processor stability and performance would be affected.

A processor scheduling algorithm acts as the organizer, coordinating the CPU's workload (Stallings, 2009 P. 406). The memory manager is tightly coupled with the processor scheduling algorithm, working to flow instructions to the CPU in an orderly fashion. These applications work two sides of a complex equation, which results in determining a weighted, value-based, decision matrix. This matrix is intended to render the right instructions, at the right time, with the least amount of idle time as possible into main memory based on an assigned priority system. Windows uses a priority matrix and a resource algorithm to determine which processes need to

be executed in order to ensure the least amount of lost clock cycles. Processes are assigned one of six priorities; Real-time, High, Above Normal, Normal, Below Normal, and Low (Microsoft Inc., 1995). Within these levels different kernel and user processes are then subcategorized by the algorithm which executes based on the weighted highest value of the ready processes (figure 1). Kernel level processes can obtain a rating priority between sixteen and thirty one while user level processes can obtain values between zero and fifteen (Bettati, 2009).

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

figure 1

The role of the processor is to execute instructions. The operating system is designed to fulfill that requirement in the most efficient method possible. Using a small program called the Dispatcher, which loads into main memory at time of boot, the dispatcher acts are the gatekeeper between the physical hardware and the instructions or processes requesting access (Stallings, 2009 p. 111). The dispatcher is designed to evaluate the immediacy of processes, their ready states, and order their execution in a manner that best utilizes the processors capabilities while maintaining stability and reducing wait time. The dispatcher controls the execution of these concurrent processes through a system of interleaving their execution. This prevents unnecessarily causing delays while waiting for a dependent process and resources to become

available (Stallings, 2009 p. 114). This is accomplished by monitoring process states within the dispatcher. In addition to process ready states, other resources such as memory pointers, data structures, I/O states are monitored (Stallings, 2009 p. 117-118). The goal of the dispatcher is to attempt to prevent a hang state resulting from a wait for resources. The dispatcher operates on a hierarchy system managing top tier process and their child threads, ensuring their status and resource allocations are monitored (Stallings, 2009 p. 161). The dispatcher is able to accomplish these functions because it has privileged access to the processor.

Clearly all processes and threads cannot run as priority. Therefore a class system exists to separate privileged (kernel) calls from regular (user) calls and sandbox them for security and stability. Privileged level access is reserved for the kernel level processes. In Windows the kernel, also known as the "NT Exec" and is comprised of a series of generic objects that handle object management, process management, virtual memory management, a security reference model management, and Input / Output (I/O) management (Ft Lewis College, nd). This keeps the objects very small and speeds execution (figure 2). These executive or superclass objects are tracked and managed by the dispatcher based on special attributes and execute as a higher priority than user level objects. Some special policies related to Kernel level objects include they cannot be paged into secondary memory or interrupted. This is in sharp contrast to User level objects. User level objects can be preempted or killed by a kernel level process.

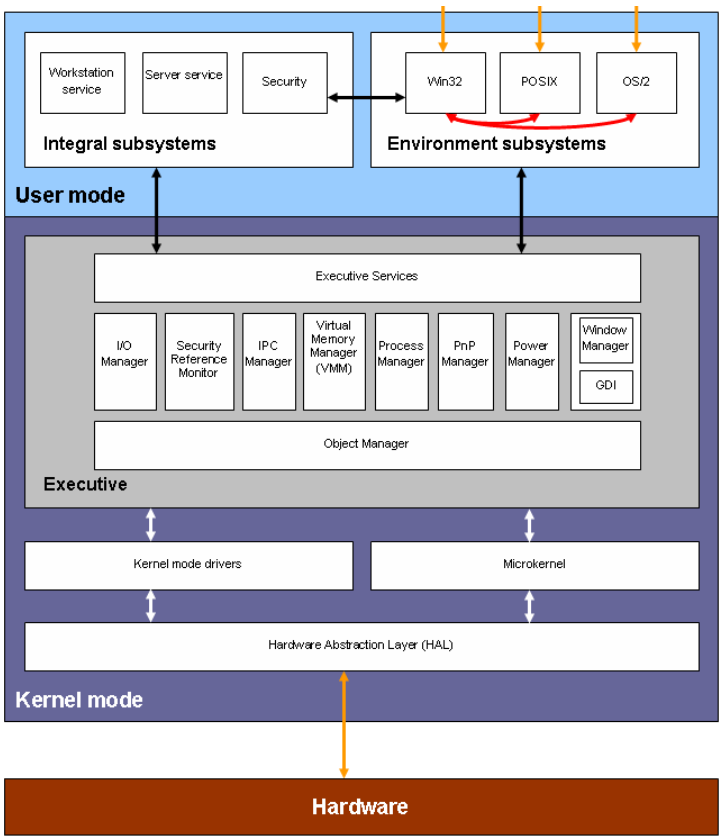

figure 2

Since computers were first constructed, one of the primary concerns was how to squeeze the most processing capability possible out of the system within the shortest period of time. This was driven by both cost and processing time availability (Stallings, 2009 p. 55). Initially computers only had one type of memory and all instructions were executed using a linear execution stream, first in, first out. Then came the advent of disk drives and output devices. It became very clear that these slower input / output (I/O) interfaces caused lost processing capability (Stallings, 2009 p. 62). As computers evolved scientists began to build subsystems to move these time intensive processes off the processor and into secondary memory and systems.

There are two types of memory, Main Memory and Secondary Memory (Stallings, 2009 p. 315). Main Memory is the only place in which the processor can receive instructions and data

in which to execute (Kozierok, 2001). Main memory is faster than secondary memory (Stallings, 2009 p. 315). Whenever possible, a system's main memory should operate at or faster than the processor clock speed. These aides in the prevention of system bottlenecks (OEMPCWorld Inc., nd). Because of the speed requirements associated with main memory, it is also more expensive than secondary memory. Main memory is organized in a single dimension address space, referred to as Linear Organization (Stallings, 2009 p. 314). Main memory is also volatile and does not retain information when power is withdrawn (Stallings, 2009 p. 315). These limitations dictate that main memory is never used for long-term storage; this task is relegated to secondary memory. When the processor call for data that is not in main memory, this is called a page fault, which is a processing inefficiency (McCraw, nd). If the information required by the processor is in secondary memory, the computer must use a system called Memory Management to retrieve the information, move it into main memory, and single an interrupt so the processor can attempt to re-execute the instruction (Stallings, 2009 p. 315).

Secondary memory is vastly different from main memory both in location and characteristics. Secondary memory is less expensive than main memory, often costing less than one U.S. dollar per megabyte. Its storage capacities can be as large at terabytes on a single drive (DEW Associates Corp., 2003). It is slower than main memory because of the requirement for Input/Out (I/O) operations across the system bus (OEMPCWorld Inc., nd). These factors combined make secondary memory the correct choice for long-term storage and short-term stack storage (Kozierok, 2001). Secondary memory also allows for an extension of main memory through a process called Virtual Memory (HowStuffWorks.com, 2001). Virtual Memory leverages storage space on a slower secondary memory device (hard drive) to put information not immediately required by the processor for execution. This allows the data to be made

available in a more orderly, on demand fashion to expediting it pop back onto the stack when required. Secondary memory does come with a down side; it's slower than main memory so the system must manage its use expeditiously to prevent system bottlenecks (Stallings, 2009 p. 312). In Windows virtual memory space is allocated using a process called paging (Stallings, 2009 p. 315). The NT Virtual Memory Manager (VMM) is the mechanism in Windows that allocates data into the paged memory (Friedman, nd). Paging divides a set space of secondary memory into frames of equal size. These frames are then loaded into main memory from secondary memory when that process is preparing to execute (Stallings, 2009 p 316). The paging size can be set automatically by windows at a rate of one and one half times the size of the physical memory or manually adjusted by an administrator.

The movement of information between main and secondary memory is a complex process that requires predictive management algorithms and constantly updated memory pointer management. Without these mechanisms, it is impossible to determine when it is efficient to move information between the two memories. Windows uses a hybrid process consisting of clustered demand paging to accomplish the fetching and the clock algorithm for replacement (Khetan, 2002). In clustered demand paging, windows brings between two and eight associated pages into memory. The clock algorithm uses a circular layout, setting an additional bit for each frame, initially at zero. As the memory manager rotates in a circular fashion, similar to the First in First Out (FIFO) memory management algorithm, it checks the usage bit. If the block is used during the cycle the bit is set to a value of one, if not it remains as zero. Blocks remaining as zeros are marked safe for swap to secondary memory. The replacement algorithm must be properly implemented and tuned to avoid thrashing; the unnecessary delay and writing to and from secondary memory (Stallings, 2009 p. 349). The combination of processor scheduling and

memory management complements each other. The result is the attempted avoidance of conditions such as process deadlocks, thrashing, and page faults, which prevent efficient execution.

A deadlock is a state where one or more processes are awaiting for a resource encumbered by different process, dependent upon the first. There are four conditions that cause deadlocks. They are Mutual Exclusion, No Preemption, Hold and Wait, and Circular Wait. This is best visualized in the Circular Wait deadlock (figure 3) which provides a clear example of how these conditions cause hang states (Stallings, 2009 p. 273). A Circular wait condition exists when a series of processes, which are processed in linear order are in a deadlock condition due to resource unavailability. In this situation the needed resource is held by the next process in line (Coffman, 1971). For example, Process 1 is waiting for Resource A. Resource A is held by Process 2, which is waiting for Resource B, which is held by Process 1. In this condition, none of the resources can be executed and the deadlock condition cannot be resolved without a broker mechanism to establish priority and terminate the appropriate deadlock. The issue of deadlocks is further complicated on multicore / multiprocessor systems where resource and I/O management becomes infinitely more complex.

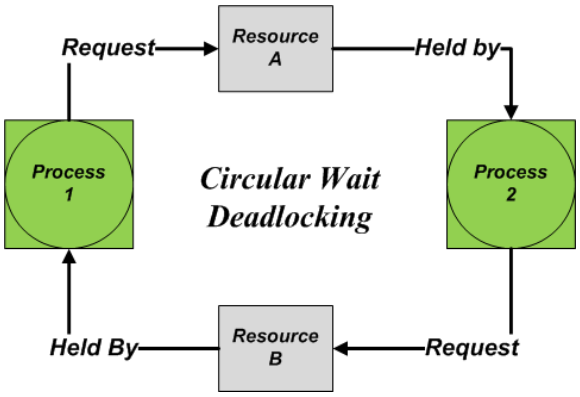

figure 3

Input / Output (I/O) Management is the process of monitoring the status of secondary memory and peripheral devices that are leveraged by the system. These devices execute tasks much slower than the CPU and as a result can cause bottlenecks in processing. When the processor requires support from an I/O device, the I/O manager is queried, the request is placed in the I/O stack, and the I/O manager places an interrupt request to the processor telling it to move onto another ready state task (Ft Lewis College, nd). The I/O manager determines the availability of an I/O device and monitors that state of the task (e.g. a secondary disk read action) and communicates its completion to the processor. This communication is in the form of an interrupt request, notifying the processor the required device is ready to complete the tasking. The offload of these requires to manage and monitor these slower peripherals improves process execution and processor efficiencies. The most common I/O requests are video output to monitors, printing, and file system access.

A file system is integrated into the operating system. It is used as the mechanism for determining how files are stored and located on a physical disk or removable media (Stallings, 2009 p. 556). File systems create a cross-walk between a friendly "File Name" and a unique binary identifier used to access the data from the physical disk (Stallings, 2009 p. 563). Files when stored have key bits of information called attributes that associate additional information used by both the file and operating systems (Stallings, 2009 p. 562). The file system keeps track of where files begin and end by assigning them a physical location on the storage media using a hierarchical addressing scheme consisting of partitions, sectors and clusters to formulate a file pointer (Microsoft Inc, 2003). The file locations along with other attributes are stored in the New Technology File System (NTFS) Master File Table and perform the function of an index of

the file system, providing the file pointer locations (Fact-index.com, nd). In Windows, NTFS leverages these attributes to determine if access to the file is allowed based on its security matrix.

The File system is responsible for the placement, location, and allocation of space for files. Windows beginning with NT server version 3.51, Microsoft introduced a secure file system called NTFS. This new system solved many of the issues related to the granular control or file access that were present under the current versions of the Disk Operating System (DOS). Under DOS, file security could not be applied below the folder level (Ft Lewis College, nd). A technology was required as computers became more dependent on networked environments to protect, share, and control vast numbers of networked resources, resulting in the development of NTFS. In order for NTFS to apply this granular control the volume that stores the files must be formatted as an NTFS partition. This formatting extends the placeholders for required additional security descriptors for files attributes within the Master File Table (MFT) and places hooks in the files which are interpreted by the Windows Security Reference Monitor (figure 2). The Security Reference Monitor implicitly executes policies regarding how files are accessed based on an access control lattice. NTFS uses a Discretionary Access Control (DAC) model for file access. Although some access controls are initially set by the operating system, an administrator has the ability to modify permissions on any file access they have administrative control. Under NTFS, when a file access is attempted, the Security Reference Model reads the identity of the requesting user and the file security descriptor attributes. It then sums these descriptors together to determine the effective permissions for the requested user. If at any point in the determination process, it encounters the "deny" permission, it is implicitly enforced, regardless of any other access grants it encounters (Microsoft Inc., 2006). Because both files and users can be nested

under various security groupings, this can be complicated to troubleshoot access issues. To resolve this Windows provides a utility called CALCS.exe which can be used to display and alter files and folders accesses control lists (Microsoft Inc., 2007).

Data protection can be accomplished in many different ways. One common method is the use of a redundant array of inexpensive disks (RAID). A RAID provides the administrator a host of recovery and availability options based on the needs and criticality of the data. The most common RAID configurations are RAID levels one, five, and ten. RAID level one mirrors the data between two identically configured disks (Stallings, 2009 p. 518). If one of the disks fails, the mirror set can be broken and the valid disk used. This method is more expensive per RAID set than the other alternatives. RAID level five uses three or more disks and distributes or stripes the data in a round robin manner (Stallings, 2009 p. 522). If one of the disks in the RAID set becomes damaged, the remaining disks can rebuild the set. This is the most common RAID implementation. RAID ten (actually designed as RAID 1+0) consists of a mirrored RAID one set that is striped as a RAID zero set. This is where the designation RAID Ten "1+0" name is derived. This RAID sets requires a minimum of four drives. One set of two drives which are mirrored holding one half the striped set and a second set of two disks mirrored for the other.

On the other side of availability is the concept of clustering. In clustering, instead of having multiple disks storing data for availability, you have multiple systems acting as one (Microsoft Inc., 2003). This solution is commonly used where high availability is a requirement, such as in ecommerce solutions. A cluster is normally three or more systems, with identical software installed. Cluster member's intercommunication with each other using a secondary NIC, connected to a VLAN and normally use private IP address network. External IP traffic is

bridged to each cluster member so that packets sent to the routable address arrive to what appears to be a single host (Abdulbasir, 2010). When the cluster is initialized, one server is selected using an algorithm to hold the quorum. The quorum owner listens to the traffic and performs various load balancing and monitoring requirements in an effort to maintain high availability. If a cluster member stops responding, the member is disabled from the cluster and the load redistributed. Regardless of the efforts taken to sustain system availability, if the system is compromised, a complete rebuild will be required.

The only system free of risk is one never powered on. Risk is introduced in many ways, through unintended errors on programming, inappropriate uses of technologies, and through intentional introductions of malicious code. Malware can be introduced in a wide variety of methods. The one key indicator of malware is it produces an outcome that was not authorized by the user (Sophos, 2010). The best defense against the introduction of malware into systems is a combination of user education and automated monitoring / preventives countermeasures (e.g. anti-virus). Users must be educated what actions are appropriate and how to recognize the signs of a potential infection. Some malware can be installed without the users' awareness. This is where tools like enterprise anti-virus solutions work best. These systems provide a signature based approach to real-time scanners on the system. When a file is accessed, the scanner evaluates the file based on behavior and checksum to determine if the file is infected. This process is designed to inhibit the spread of an infection. Most malware today is successful primarily due to administrators not maintaining and patching systems properly (Sophos, 2010).

Patching systems is a requirement for the increased stability, security and availability. It is impossible for venders to produce programs that consist of thousands of lines of code without unforeseen errors. When an error is introduced into the production environment, the application

must be fixed. These fixes are called patches or partial changes to a complete application. Sometimes patches improve usability while other times they plug security holes. Regardless of the reason for the patch, it's important that all system administrators monitor the patch repositories and vender sites, identifying and installing new patches as soon as possible (based on system uptime requirements prior testing). Microsoft as well as other venders provides independent patch repositories for their products. This complicates issues, forcing administrators to check multiple sites to ensure all applications installed on the system are fully patched. Inconsistent and delayed patching practices lead to down time, lost productivity, and potential data breaches (SANS Institute, 2010). Patching systems is a basic systems administrator's responsibility, which must be monitored for the overall networked ecosystems health.

We have reviewed the Windows Operating System from the perspective of how the operating system provides the services necessary to sustain operability. We have examined the relationship between the software instructions and the hardware execution, to include the complex transactional calculations that occur to maintain availability. Finally we have reviewed the steps necessary to provide sustained data and system availability. Without such protections these system would not be able to process data. It is my sincerely hope this paper has provided you a clearer understanding of this often overlook mechanisms.

References:

Abdulbasir, I. (2010, January 26). *Bridging two Nics in windows server 2003 R2*. Retrieved May 22, 2010, from http://networking.ittoolbox.com/groups/technical-functional/networkadmin-l/bridging-two-nics-in-windows-server-2003-r2-3261320

Bettati, R. (2009). *Windows NT and Real-Time*. Retrieved May 22, 2010, from CPSC-663: Real-Time Systems: http://faculty.cs.tamu.edu/bettati/Courses/663/2009C/Slides/RTWindows.pdf

Coffman. (1971). *Necessary and Sufficient Deadlock Conditions*. Retrieved April 13, 2010, from http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/deadlockCondition.htm

DEW Associates Corp. (2003). *Hard Drive Size Limitations and Barriers*. Retrieved April 23, 2010, from http://www.dewassoc.com/kbase/index.html

Fact-index.com. (nd). *File Allocation Table.* Retrieved May 6, 2010, from http://www.fact-index.com/f/fi/file_allocation_table.html

Friedman, M. B. (nd). *Windows NT page replacement policies*. Retrieved MAY 22, 2010, from http://www.demandtech.com/Resources/Papers/WinMemMgmt.pdf

Ft Lewis College. (nd). *The Windows NT Architecture* . Retrieved May 22, 2010, from http://scilnet.fortlewis.edu/tech/NT-Server/architecture.htm

HowStuffWorks.com. (2001, July 20). *What is virtual memory*. Retrieved April 2010, 2010, from http://computer.howstuffworks.com/question684.htm

Khetan, G. (2002, December 16). *Comparison of Memory Management Systems of BSD, Windows, and Linux*. Retrieved May 22, 2010, from University of California at Los Angeles Computer Science Dept: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97

Kozierok, C. M. (2001, April 17). *System Memory*. Retrieved April 24, 2010, from http://www.pcguide.com/ref/ram/index.htm

McCraw, T. (nd). *Virtual Memory Page Replacement Algorithms*. Retrieved April 23, 2010, from http://people.msoe.edu/~mccrawt/resume/papers/CS384/mccrawt_cs384_virtual.pdf

Microsoft Inc. (2003, March 28). *NTFS Technical Reference.* Retrieved May 6, 2010, from Technet: http://technet.microsoft.com/en-us/library/cc758691(WS.10).aspx

Microsoft Inc. (2007). *Cacls* . Retrieved May 22, 2010, from Technet: http://technet.microsoft.com/en-us/library/bb490872.aspx

Microsoft Inc. (2006, November 1). *How To Configure Security for Files and Folders on a Network (Domain) in Windows 2000*. Retrieved May 22, 2010, from Support: http://support.microsoft.com/kb/301195

Microsoft Inc. (1995, June 29). *Real-Time Systems and Microsoft Windows NT*. Retrieved May 22, 2010, from MSDN: http://msdn.microsoft.com/en-us/library/ms810433.aspx

Microsoft Inc. (2003, March 28). *What Is a Server Cluster*. Retrieved May 22, 2010, from Technet: http://technet.microsoft.com/en-us/library/cc785197(WS.10).aspx

OEMPCWorld Inc. (nd). *Memory Access vs CPU Speed*. Retrieved April 24, 2010, from http://www.oempcworld.com/support/Memory_Access_vs_CPU_Speed.htm

SANS Institute. (2010, May 20). *Penetration Testing*. Retrieved May 20, 2010, from SANS InfoSec Reading Room: http://www.sans.org/reading_room/whitepapers/testing/

Sophos. (2010). *Security Threat Report 2010.* Retrieved April 26, 2010, from http://www.sophos.com/sophos/docs/eng/papers/sophos-security-threat-report-jan-2010-wpna.pdf

Stallings, W. (2009). *Operating Systems.* Upper Saddle River, NJ: Pearson Prentice Hall.