WINDOWS NT:

HISTORY:

1. 1981, Microsoft begins development of the Interface Manager (subsequently renamed Microsoft Windows). Although the first prototypes used Multiplan and Word-like menus at the bottom of the screen, the interface was changed in 1982 to use pull-down menus and dialogs, as used on the Xerox Star. Microsoft finally announced Windows in November 1983, with pressure from just-released VisiOn and impending TopView. This was after the release of the Apple Lisa, and before Digital Research announced GEM, and DESQ from Quarterdeck and the Amiga Workbench, or GEOS/GeoWorks Ensemble, IBM OS/2, NeXTstep or even DeskMate from Tandy.

   Development was delayed several times, and the Windows 1.0 hit the store shelves in November 1985. The selection of applications was sparse, however, and Windows sales were modest.

2. 1985, Microsoft releases an operating system called MS-NET, which was an attempt to compete with the most popular operating system at the time (Novell's NetWare). It was really just a version of 3com's OpenServer NOS (network operating system) that it licensed and renamed Microsoft's LAN Manager. But Microsoft really wanted to develop a new technology, peer-to-peer networking, which means that clients share files and printers like servers, therefore, a small network can exist without a giant server. LAN Manager used IBM's NetBEUI, which IBM released in 1985. NetBEUI is a fast and efficient protocol, but it is non-routable, which means it cannot be easily routed to other network segments, but is not impossible.

3. Mid 80s, Microsoft and IBM develop OS/2 (in assembly). It eliminated the 640k conventional memory barrier and a new file system that would allow long filenames and a fault tolerance.

4. 1987, Windows 2.0 is introduced, providing significant usability improvements. With the addition of icons and overlapping windows, Windows became a viable environment for development of major applications.

Windows/386 was released later in 1987. While it was functionally equivalent to its sibling, Windows/286, in running Windows applications, it provided the capability to run multiple DOS applications simultaneously in extended memory.

5. 1988 Microsoft decides to develop its own “new technology” (NT) operating system. An OS that supports OS/2 and POSIX.

6. 1988 Microsoft hires Dave Culter (architect of VAX/VMS).

7. 1989, Microsoft drops OS/2 and guides NT to be Windows compliant.

8. 1990, Windows 3.0 released. A complete overhaul of the Windows environment. With the capability to address memory beyond 640K and a much more powerful user interface, independent software vendors started developing Windows applications with vigor.

9. 1992, Windows 3.1 released providing significant improvements to Windows 3.0.

10. Windows 3.11, which added no new features but corrected some existing, mostly network-related problems, replaced Windows 3.1 in later 1992.

11. Windows for Workgroups 3.1, released in October, 1992, was the first integrated Windows and networking package offered by Microsoft. It provided peer-to-peer file and printer sharing capabilities highly integrated into the Windows environment. The simple-to-use-and-install networking allows the user to specify which files on the user's machine should be made accessible to others. The files can then be accessed from other machines running either Windows or DOS. Windows for Workgroups also includes two additional applications: Microsoft Mail, a network mail package, and Schedule+, a workgroup scheduler.

12. 1993, Windows NT 3.1 released.

13. 1994, Windows NT 3.5 released. 3.5 provides OLE 2.0, improved performance and reduced memory requirements.

Early versions of NT (esp. 3.51) were known to be very reliable systems. However, they were not very performance oriented.

14. 1995, Windows NT 4.0, ("Cairo") 4.0 was Microsoft's project for object-oriented Windows. **NT 4.0 moved some of the user-level functionality into the kernel in order to improve performance.** Virtual desktops also introduced (screen saver, user screen, and ctrl-alt-delete).

15. 1995, Windows 95 released. A 32-bit system providing full pre-emptive multitasking, advanced file systems, threading, networking and more. Included MS-DOS 7.0, but took over from DOS completely after starting. Also included a completely revised user interface.

16. 1998, Windows 98 released. Integrated Web Browsing gives your desktop a browser-like interface. You 'browse' everything, including stuff on your local computer.

17. 2000, Windows NT 5.0 (Windows 2000) released. Included a host of new features including the integration of Internet Explorer 4.0 into the operating system.

18. 2000, Windows Me (Millenium Edition) released. Me boasts some enhanced multimedia features, such as an automated video editor and improved Internet plumbing. But unlike Microsoft's Windows 2000 OS which offers advanced security, reliability, and networking features, Windows Me is just an upgrade to the DOS-based code on which previous Windows versions were built.

19. 2001, Windows XP. Ballyhooed as a whole new kind of Windows for consumers. Under the hood, it contains the same 32-bit kernel and driver set from Windows NT / 2000. However, they do claim that it is more secure and more stable. There were 64-bit versions of XP also (originally for the Itanium, or "Itanic").

20. 2006, Windows Vista. While Vista was horrible (worse than ME?), it was used to introduce many new features, such as an image based file system (a serialized copy of the entire state of the computer system), Hyper-V, and MinWin (a low memory web server). Vista also had the largest memory footprint ever.

21. 2009, Windows 7.

22. 2012, Windows 8 had 48 bits of memory space (terabytes).

23. 2015, Windows 10. Windows 10 has a number of design changes to it. Windows 10 also supports biometrics for logging on. Windows 10 also provides better support to run Linux from without a VM.

<u>DESIGN GOALS:</u>

- The unique design of Windows-NT can be attributed to:

    1. It's a Microsoft product (commercial & proprietary).

- The design goals where:

    1. Extensibility – The OS has a layered design. The kernel uses dynamically loadable device drivers (modules). Layers above the kernel provide interfaces into the different emulated environments (DOS, Windows, and POSIX). All three help to define an OS that is easily adapted to new hardware.

    2. Portable – The majority of the code is written in C or C++. All processor specific code is isolated in a DLL. To move to a different platform requires a different DLL and a recompile.

    3. Reliability – NT is designed to resist defects and attacks by using hardware protection for virtual memory and software protection for OS resources. The native file system can repair/recover itself from many finds of errors after a system crash. NT has a C2 security certification.

    4. Compatibility – NT provide source code level compatibility (code should only have to be recompiled and not modified). NT can also run binaries from DOS and Windows. On non-intel platforms, NT provides a x86 emulator. NT's subsystems also provide support for multiple file systems.

    5. Performance – Expect for the kernel, threads in upper layers can be preempted by higher priority threads. NT also supports SMP (though not as well as UNIX).

6. International use – NT provides support for national language support (NLS).

SYSTEM COMPONENTS:

- NT divides the system into 3 layers: the hardware abstraction layer, the kernel layer, and the user layer.

1. HARDWARE ABSTRACTION LAYER (HAL):

- A layer of software that hides hardware differences from the kernel. HAL also provides the support for SMP.

2. KERNEL LAYER:

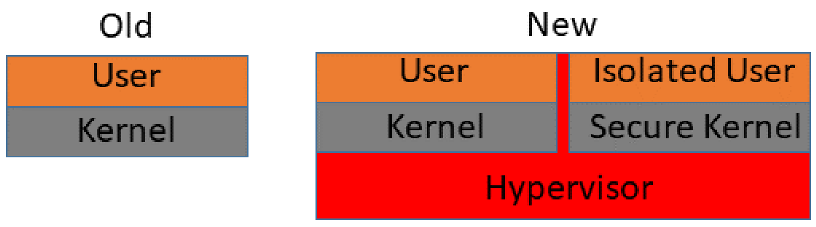
- The kernel provides functionality for thread scheduling (NT supports threaded processes), interrupt handling, processor synchronization, and power management.

  1. The kernel is never paged out and never preempted.

  2. Processes/threads can have an affinity for a specific processor (on an SMP machine).

- The kernel layer includes an executive layer, which provides services for virtual memory management, process management, I/O, interprocess communication, and security.

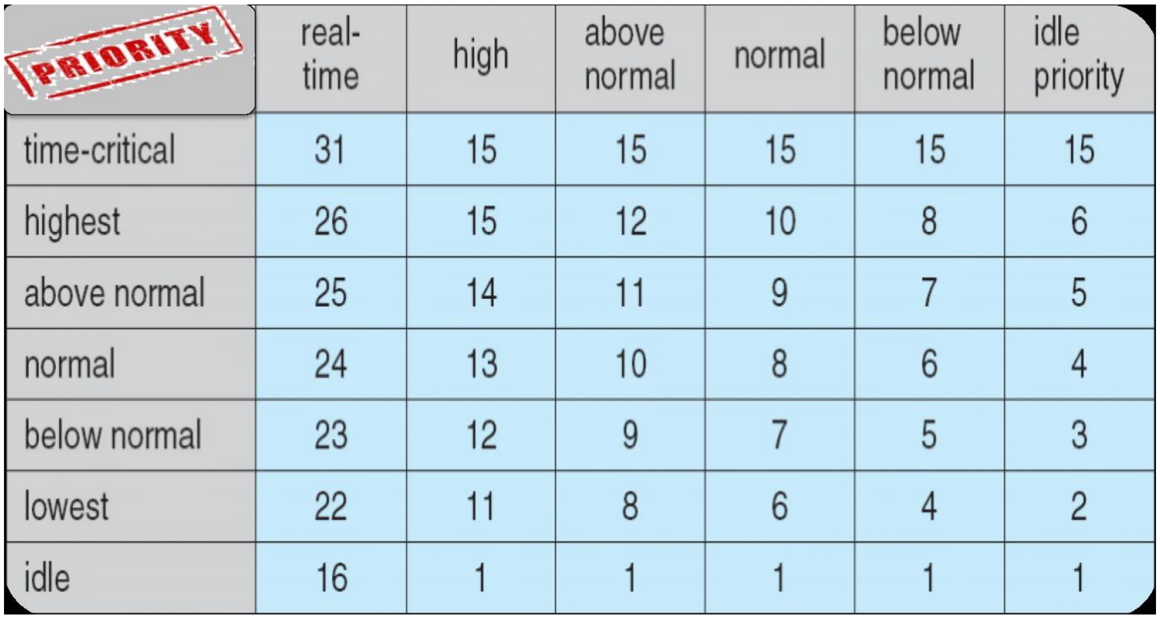
- In Windows 10, the kernel is now a secure kernel. In a sense this provides a second operating system to run things on. Windows 10 also supports containers (a standard unit of software that packages up code and dependencies so the application runs reliably from one computing environment to another). Windows 10 has an improved TCP/IP stack.



- Process/thread scheduling:

1. The kernel uses a 32 level priority scheme to determine the order of process/thread (I'll refer to both simply as processes) execution.

2. Priorities are divided into classes:



PRIORITY	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

3. The scheduler uses a round-robin queue for each priority level and scans through the queues (from high to low priority) looking for the first process ready to run. If no ready process is found an "idle" thread is run.

4. When the time quantum is up, the process is interrupted.

5. When I/O is required, the process is interrupted.

6. A higher priority process will interrupt a lower priority process (I.e. Priority Inversion can occur).

7. Each thread has a two priorities. Initially, a thread's dynamic priority is the same as its base priority. The system can boost and lower the dynamic priority, to ensure that it is responsive and that no threads are starved for processor time. The

system does not boost the priority of threads with a base priority level between 16 and 31. Only threads with a base priority between 0 and 15 receive dynamic priority boosts.

8. When a process that has NORMAL_PRIORITY_CLASS is brought to the foreground, the scheduler boosts the priority class of the process associated with the foreground window, so that it is greater than or equal to the priority class of any background processes. The priority class returns to its original setting when the process is no longer in the foreground.

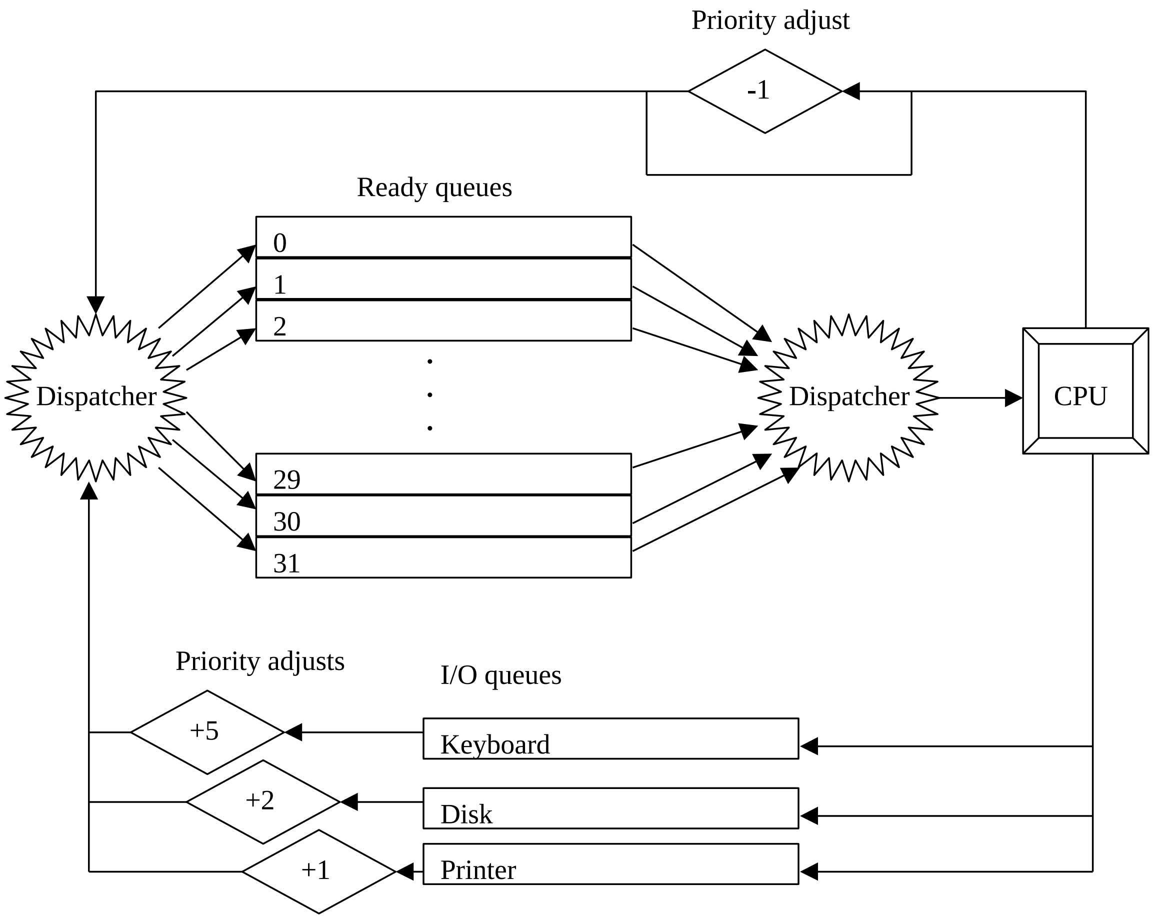
9. When a window receives input, such as timer messages, mouse messages, or keyboard input, the scheduler boosts the priority of the thread that owns the window.

10. When the wait conditions for a blocked thread are satisfied, the scheduler boosts the priority of the thread. For example, when a wait operation associated with disk or keyboard I/O finishes, the thread receives a priority boost.

11. After raising a thread's dynamic priority, the scheduler reduces that priority by one level each time the thread completes a time slice, until the thread drops back to its base priority. A thread's dynamic priority is never less than its base priority.

- Like UNIX and LINUX, NT favors I/O bound processes and "punishes" CPU bound processes.

- Process/thread scheduling (cont):



- Interrupt management:

  1. The kernel manages both interrupts and exceptions (like software interrupts - divide by zero, etc).

  2. When kernel exceptions occur, the kernel calls the appropriate routine to handle the exception. If no routine is found, the system dies ("blue screen of death").

  3. When user-level exceptions occur, the kernel attempts to find a method to handle the exception (a running debugger, an existing exception handler, or terminate the process).

  4. When interrupts occur, the kernel calls the appropriate routine to handle the interrupt (a device driver or a kernel-level routine).

- Power management (if supported by BIOS):

1. The power-fail interrupt has the second highest priority (highest priority is the machine check or bus error interrupt) and notifies the OS whenever a power loss is detected. About all this interrupt can do is record the fact that the power is going down. On reboot this info is used to reset the state of the device drivers.

   This interrupt is especially useful on UPS backed systems where it can be used to prevent a process from going into its critical section when the machine is running on batteries.

- Memory management (executive layer):

  1. Uses a page based scheme (page size of 4KB). Pages not in physical memory are stored on disk (in a paging file - i.e. a swap file).

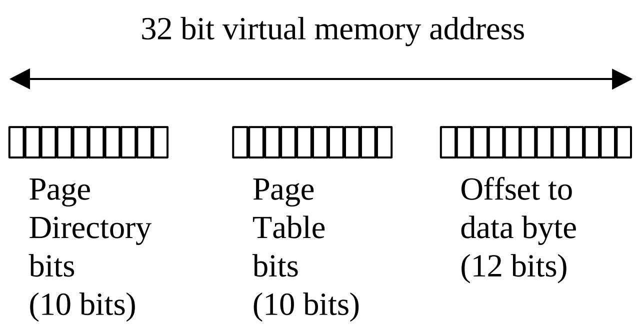
  2. The memory manager uses a 2 step process for allocating memory. The first step is to reserve the required memory. The second step is to commit that memory to the process (NT can limit the amount of memory that a process reserves/commits).

  3. Privileged processes can lock pages in memory (preventing them from being paged).

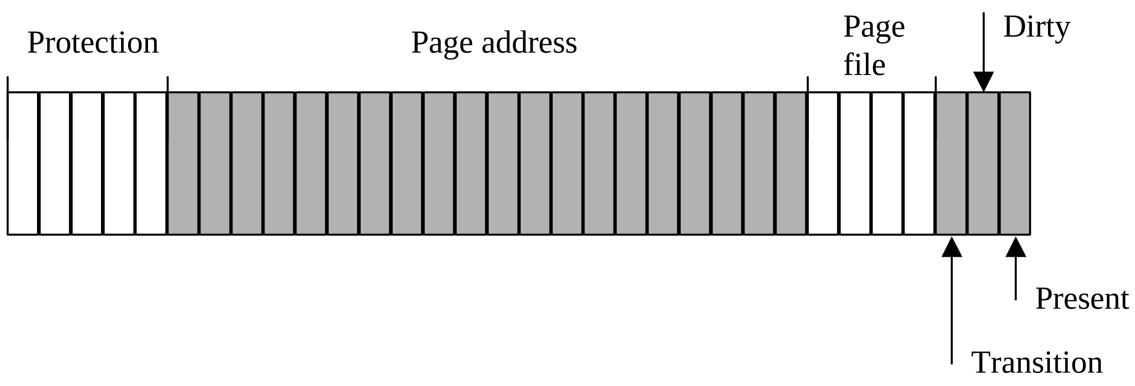
  4. Two processes can share memory via a "view". A process can control the size of the view, have the view be backed by disk, and set various levels of protection on the view.

  5. The virtual memory manager uses a 3 level tree to allocate system memory. The root level (page directory) points to 1024 middle-level nodes (page tables) which point to 1024 4 KB pages each.

  6. Virtual memory addressing is via a 32 bit word which is divided into 3 sections:


32 bit virtual memory address
Page Directory bits (10 bits)
Page Table bits (10 bits)
Offset to data byte (12 bits)

7. Physical memory addressing is done by concatenating 20 bits from the PTE and page offset (of the virtual memory address) with an additional 12 bits (which provide protection, page file, and state information).



8. The virtual memory manager keeps track of all pages of physical memory in a page-frame database (1 entry/page frame). Frames are linked to form a list of free or zeroed pages. The entry points to the PTE that points to the page frame.

   I.e. If you think of it this way: each page-frame database entry (directory entry) points to the PTE (File allocation table) which points to the page frame (disk block). The virtual memory manager looks like the FAT method for disks.

9. When a page fault occurs (the page is not in memory), the virtual memory manager loads the required page into the first free page. However, for performance concerns the virtual memory manager will also prefetch a few adjacent pages. Since we may want to reuse those pages again, they are kept in a cache (standby list) for faster reuse. Pages that

need to be written to the drive are stored in a modified list for batch writes.

10. If there are no free pages left, NT uses a per process FIFO page replacement policy (a local page replacement policy - reduces thrashing). Initially processes are given 30 pages. NT periodically "locks-out" an old page from processes. If the process continues without generating a page fault (for the "locked-out" page), the processes looses the page (which is added to the free list). This tends to reduce memory "waste".

11. In Windows 10 pages are compressed in memory making it possible to store more pages in memory. 10 also uses a “RAM drive” to store “swapped out” pages (still compressed). The real hard drive is also used (when programs are shut down?) to store pages (still compressed). The intent of this was to save SSDs. Memory is 100,000 x faster than a HD and 50x faster than an SSD. Page faults have been shown to reduce the life of an SSD, so by compression pages, far fewer writes to the HD occur.

- I/O management (executive layer):

  1. The I/O manager is responsible for all I/O (file systems, device drivers, and cache).

  2. The I/O manager keeps track of which file systems are loaded.

  3. The I/O manager manages the centralized (write-back) cache facility. The cache size changes with the amount of free memory and uses a block size of 256KB. In an effort to predict future requests (to provide prefetching), the cache manager keeps track of recent requests. To keep the disk synced with the cache, every 4-5 seconds the cache (blocks marked with a dirty bit) is flushed to the disk. When free cache space gets low, the cache manager will block processes from writing to the cache until it has been able to sync the cache with the disk.

3. USER LAYER:

- The environmental subsystems are user-mode processes layered over the native NT (win32) executive services to enable NT to run programs developed for other OSs.

  1. DOS is implemented by a virtual dos machine (VDM - based on MSDOS 5.0). VDM, which is a user-level process, is paged like any other process. VDM provides emulation of a 486 complete with the BIOS and "int 21" software interrupt services. DOS is not a multitasking environment and some applications that are CPU hogs will be forced by NT to behave according to NT's rules. These applications frequently do not operate correctly.

  2. Windows3.x is implemented by another VDM (which includes some Windows3.x kernel routines and windows interfaces). The 16 bit API is not fully implemented (so some Windows3.x programs will not run) and multiple Windows3.x applications can interfere with each other (just like in the "real" Windows3.x).

  3. POSIX is only partially implemented. Printing, networking, and graphics are not directly supported if at all.

  4. OS/2 applications are not fully supported and must be run as DOS apps.

FILE SYSTEM:

- FAT (16 bit) has no protection mechanism, a 2 GIG size limit, and a lot of internal fragmentation.

- FAT (32 bit) solved the size and internal fragmentation problems, but still has no protection mechanism.

- NTFS (NT file system)

  1. NTFS supports symbolic links.

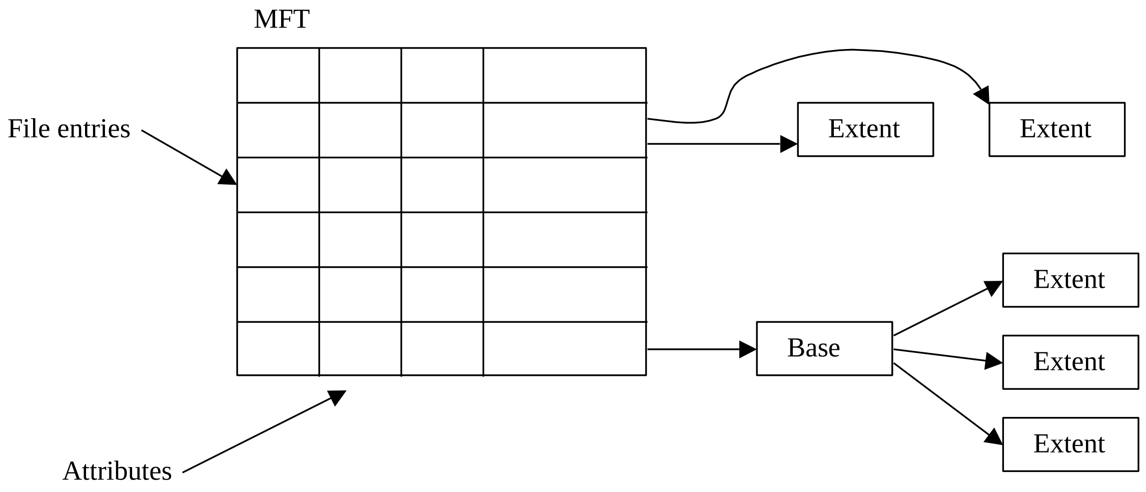
  2. Based on volumes (which may span multiple disks).

  3. Uses groups of sectors (clusters) as the allocation unit. Cluster size will vary with the volume size (bigger volume, bigger cluster size).

  4. Clusters are numbered from the disk beginning to end.

5. Files in NTFS are simple byte streams (as in DOS or UNIX). They are structured and contain attribute data (file name, creation date/time, protection, etc). These attributes can be read, deleted, changed, written independently of the data portion of the file.

6. All NTFS files have entries in the master file table (MFT). Small attributes (all but the data generally, except for very small files) are stored in the MFT itself (resident attributes). Large attributes are stored as *extents* on the disk (nonresident attributes).



7. NTFS directories are stored as B+ trees. B+ trees are balanced binary trees, which also have a sequential set of pointers through the leaf nodes.

8. File names (as well as some attribute data) are stored at the leaf nodes. So, to do a directory listing all one has to do is a sequential search through the B+ tree.

9. NTFS takes a database approach to maintain file system consistency. All file system updates (updates to file system files, not user files) are considered to be *transactions*. And like a database transaction, NTFS transactions record the redo and undo information in a log. After the change has been completed a *commit* is written to the log. If the system should fail, uncommitted changes to the file system can easily be redone or, if necessary, undone. About every 5 seconds

a checkpoint in written to the log (logged changes older than the checkpoint can be discarded). Whenever the system is restarted after some failure, the file system recovering is performed automatically.

10. NTFS supports raid 0, raid 1, and raid 5.

11. NTFS also supports optional on-the-fly compression.

IN A NUTSHELL:

- An OS designed with PC performance in mind and user interactivity.