OPERATING SYSTEMS DESIGN

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the operating system
- Understand the Program execution
- Understand the Interrupts
- Understand the Supervisor mode
- Understand the Memory management
- Understand the Virtual memory
- Understand the Multitasking

Definition/Overview:

An operating system: An operating system (commonly abbreviated to either OS or O/S) is an interface between hardware and applications; it is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. The operating system acts as a host for applications that are run on the machine.

Program execution: The operating system acts as an interface between an application and the hardware.

Interrupts: Interrupts are central to operating systems as they provide an efficient way for the operating system to interact and react to its environment.

Supervisor mode: Modern CPUs support something called dual mode operation. CPUs with this capability use two modes: protected mode and supervisor mode, which allow certain CPU functions to be controlled and affected only by the operating system kernel. Here, protected mode does not refer specifically to the 80286 (Intel's x86 16-bit microprocessor) CPU feature, although its protected mode is very similar to it.

Memory management: Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs.

Key Points:

1. An operating system

An operating system (commonly abbreviated to either OS or O/S) is an interface between hardware and applications; it is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. The operating system acts as a host for applications that are run on the machine. As a host, one of the purposes of an operating system is to handle the details of the operation of the hardware. This relieves application programs from having to manage these details and makes it easier to write applications. Almost all computers, including handheld computers, desktop computers, supercomputers, and even video game consoles, use an operating system of some type. Some of the oldest models may however use an embedded operating system, that may be contained on a compact disk or other data storage device . Operating systems offer a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls. By invoking these interfaces, the application can request a service from the operating system, pass parameters, and receive the results of the operation. Users may also interact with the operating system with some kind of software user interface (UI) like typing commands by using command line interface (CLI) or using a graphical user interface (GUI, commonly pronounced gooey). For hand-held and desktop computers, the user interface is generally considered part of the operating system. On large multi-user systems like UNIXand Unix-like systems, the user interface is generally implemented as an application program that runs outside the operating system. (Whether the user interface should be included as part of the operating system is a point of contention.) Common contemporary operating systems include Microsoft Windows, Mac OS, Linux, BSD and Solaris. Microsoft Windows has a significant majority of market share in the desktop and notebook computer markets, while servers generally run on Linux or other Unix-like systems. Embedded device markets are split amongst several operating systems.

2. Program execution

The operating system acts as an interface between an application and the hardware. The user interacts with the hardware from "the other side". The operating system is a set of services which simplifies development of applications. Executing a program involves the creation of a process by the operating system. The kernel creates a process by assigning memory and other

resources, establishing a priority for the process (in multi-tasking systems), loading program code into memory, and executing the program. The program then interacts with the user and/or other devices performing its intended function.

3. Interrupts

Interrupts are central to operating systems as they provide an efficient way for the operating system to interact and react to its environment. The alternative is to have the operating system "watch" the various sources of input for events that require action -- not a good use of CPU resources. Interrupt-based programming is directly supported by most CPUs. Interrupts provide a computer with a way of automatically running specific code in response to events. Even very basic computers support hardware interrupts, and allow the programmer to specify code which may be run when that event takes place. When an interrupt is received the computer's hardware automatically suspends whatever program is currently running, saves its status, and runs computer code previously associated with the interrupt. This is analogous to placing a bookmark in a book when someone is interrupted by a phone call and then taking the call. In modern operating systems interrupts are handled by the operating system's kernel. Interrupts may come from either the computer's hardware or from the running program. When a hardware device triggers an interrupt the operating system's kernel decides how to deal with this event, generally by running some processing code. How much code gets run depends on the priority of the interrupt (for example: a person usually responds to a smoke detector alarm before answering the phone). The processing of hardware interrupts is a task that is usually delegated to software called device drivers, which may be either part of the operating system's kernel, part of another program, or both. Device drivers may then relay information to a running program by various means. A program may also trigger an interrupt to the operating system. If a program wishes to access hardware for example, it may interrupt the operating system's kernel, which causes control to be passed back to the kernel. The kernel will then process the request. If a program wishes additional resources (or wishes to shed resources) such as memory, it will trigger an interrupt to get the kernel's attention.

4. Supervisor mode

Modern CPUs support something called dual mode operation. CPUs with this capability use two modes: protected mode and supervisor mode, which allow certain CPU functions to be controlled and affected only by the operating system kernel. Here, protected mode does not

refer specifically to the 80286 (Intel's x86 16-bit microprocessor) CPU feature, although its protected mode is very similar to it. CPUs might have other modes similar to 80286 protected mode as well, such as the virtual 8086 mode of the 80386 (Intel's x86 32-bit microprocessor or i386). However, the term is used here more generally in operating system theory to refer to all modes which limit the capabilities of programs running in that mode, providing things like virtual memory addressing and limiting access to hardware in a manner determined by a program running in supervisor mode. Similar modes have existed in supercomputers, minicomputers, and mainframes as they are essential to fully supporting UNIX-like multi-user operating systems. When a computer first starts up, it is automatically running in supervisor mode. The first few programs to run on the computer, being the BIOS, bootloader and the operating system have unlimited access to hardware - and this is required because, by definition, initializing a protected environment can only be done outside of one. However, when the operating system passes control to another program, it can place the CPU into protected mode. In protected mode, programs may have access to a more limited set of the CPU's instructions. A user program may leave protected mode only by triggering an interrupt, causing control to be passed back to the kernel. In this way the operating system can maintain exclusive control over things like access to hardware and memory. The term "protected mode resource" generally refers to one or more CPU registers, which contain information that the running program isn't allowed to alter. Attempts to alter these resources generally cause a switch to supervisor mode, where the operating system can deal with the illegal operation the program was attempting (for example, by killing the program).

5. Memory management

Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs. This ensures that a program does not interfere with memory already used by another program. Since programs time share, each program must have independent access to memory. Cooperative memory management, used by many early operating systems assumes that all programs make voluntary use of the kernel's memory manager, and do not exceed their allocated memory. This system of memory management is almost never seen anymore, since programs often contain bugs which can cause them to exceed their allocated memory. If a program fails it may cause memory used by one or more other programs to be affected or overwritten. Malicious programs or viruses may purposefully alter another program's memory or may

affect the operation of the operating system itself. With cooperative memory management it takes only one misbehaved program to crash the system. Memory protection enables the kernel to limit a process' access to the computer's memory. Various methods of memory protection exist, including memory segmentation and paging. All methods require some level of hardware support (such as the 80286 MMU) which doesn't exist in all computers. In both segmentation and paging, certain protected mode registers specify to the CPU what memory address it should allow a running program to access. Attempts to access other addresses will trigger an interrupt which will cause the CPU to re-enter supervisor mode, placing the kernel in charge. This is called a segmentation violation or Seg-V for short, and since it is both difficult to assign a meaningful result to such an operation, and because it is usually a sign of a misbehaving program, the kernel will generally resort to terminating the offending program, and will report the error. Windows 3.1-Me had some level of memory protection, but programs could easily circumvent the need to use it. Under Windows 9x all MS-DOS applications ran in supervisor mode, giving them almost unlimited control over the computer. A general protection fault would be produced indicating a segmentation violation had occurred, however the system would often crash anyway. In most Linux systems, part of the hard disk is reserved for virtual memory when the Operating system is being installed on the system. This part is known as swap space. Windows systems use a swap file instead of a partition.

6. Virtual memory

The use of virtual memory addressing (such as paging or segmentation) means that the kernel can choose what memory each program may use at any given time, allowing the operating system to use the same memory locations for multiple tasks. If a program tries to access memory that isn't in its current range of accessible memory, but nonetheless has been allocated to it, the kernel will be interrupted in the same way as it would if the program were to exceed its allocated memory. (See section on memory management.) Under UNIXthis kind of interrupt is referred to as a page fault. When the kernel detects a page fault it will generally adjust the virtual memory range of the program which triggered it, granting it access to the memory requested. This gives the kernel discretionary power over where a particular application's memory is stored, or even whether or not it has actually been allocated yet. In modern operating systems, application memory which is accessed less frequently can be temporarily stored on disk or other media to make that space available for use by other

programs. This is called swapping, as an area of memory can be used by multiple programs, and what that memory area contains can be swapped or exchanged on demand.

7. Multitasking

Multitasking refers to the running of multiple independent computer programs on the same computer, giving the appearance that it is performing the tasks at the same time. Since most computers can do at most one or two things at one time, this is generally done via time sharing, which means that each program uses a share of the computer's time to execute. An operating system kernel contains a piece of software called a scheduler which determines how much time each program will spend executing, and in which order execution control should be passed to programs. Control is passed to a process by the kernel, which allows the program access to the CPU and memory. At a later time control is returned to the kernel through some mechanism, so that another program may be allowed to use the CPU. This so-called passing of control between the kernel and applications is called a context switch. An early model which governed the allocation of time to programs was called cooperative multitasking. In this model, when control is passed to a program by the kernel, it may execute for as long as it wants before explicitly returning control to the kernel. This means that a malicious or malfunctioning program may not only prevent any other programs from using the CPU, but it can hang the entire system if it enters an infinite loop. The philosophy governing preemptive multitasking is that of ensuring that all programs are given regular time on the CPU. This implies that all programs must be limited in how much time they are allowed to spend on the CPU without being interrupted. To accomplish this, modern operating system kernels make use of a timed interrupt. A protected mode timer is set by the kernel which triggers a return to supervisor mode after the specified time has elapsed. (See above sections on Interrupts and Dual Mode Operation.) On many single user operating systems cooperative multitasking is perfectly adequate, as home computers generally run a small number of well tested programs. Windows NT was the first version of Microsoft Windows which enforced preemptive multitasking, but it didn't reach the home user market until Windows XP, (since Windows NT was targeted at professionals.)

9. Layers of Abstraction

An abstraction layer (or abstraction level) is a way of hiding the implementation details of a particular set of functionality. Software models that use layers of abstraction include the OSI

7 Layer model for computer network protocols, the OpenGL graphics drawing library, and the byte stream input/output (I/O) model originated by UNIXand adopted by MSDOS, Linux, and most other modern operating systems. In the UNIXoperating system, most types of input and output operations are considered to be streams of bytes being read from a device or being written to a device. This stream of bytes model is used for file I/O, socket I/O, and terminal I/O in order to provide device independence. In order to read and write to a device at the application level, the program calls a function to open the device which may be a real device such as a terminal or a virtual device such as a network port or a file in a file system. The device's physical characteristics are mediated by the operating system which in turn presents an abstract interface that allows the programmer to read and write bytes from/to the device. The operating system then performs the actual transformation needed to read and write the stream of bytes to the device. Most graphics libraries such as OpenGL provide an abstract graphical device model as an interface. The library is responsible for translating the commands provided by the programmer into the specific device commands needed to draw the graphical elements and objects. The specific device commands for a plotter are different from the device commands for a CRT monitor but the graphics library hides the implementation and device dependent details by providing an abstract interface which provides a set of primitives that are generally useful for drawing graphical objects. In computer science, an abstraction level is a generalization of a model or algorithm, away from any specific implementation. These generalizations arise from broad similarities that are best encapsulated by models that express similarities present in various specific implementations. The simplification provided by a good abstraction layer allows for easy reuse by distilling a useful concept or metaphor so that situations where it may be accurately applied can be quickly recognized. A good abstraction will generalize that which can be made abstract; while allowing specificity where the abstraction breaks down and its successful application requires customization to each unique requirement or problem. Frequently abstraction layers can be composed into a hierarchy of abstraction levels. The ISO-OSI networking model comprises seven abstraction layers. Each layer of the OSI ISO networking model encapsulates and addresses a different part of the needs of much digital communications thereby reducing the complexity of the associated engineering solutions. A famous aphorism of Butler Lampson goes: All problems in computer science can be solved by another level of indirection; this is often deliberately mis-quoted with "abstraction" substituted for "indirection". Kevlin Henney's corollary to this is, "...except for the problem of too many layers of indirection."

10. Open source

Open source software (OSS) began as a marketing campaign for free software. OSS can be defined as computer software for which the human-readable source code is made available under a copyright license (or arrangement such as the public domain) that meets the Open Source Definition. This permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form. It is very often developed in a public, collaborative manner. Open source software is the most prominent example of open source development and often compared to user-generated content. A report by Standish Group says that adoption of open source has resulted in savings of about $60 billion per year to consumers.

11. Open source software vs. free software

Critics have said that the term open source fosters an ambiguity of a different kind such that it confuses the mere availability of the source with the freedom to use, modify, and redistribute it. Developers have used the alternative terms Free/open source Software (FOSS), or Free/Libre/open source Software (FLOSS), consequently, to describe open source software which is also free software. The term Open Source was originally intended to be trademarkable; however, the term was deemed too descriptive, so no trademark exists. The OSI would prefer that people treat Open Source as if it were a trademark, and use it only to describe software licensed under an OSI approved license. . There have been instances where software vendors have labeled proprietary software as open source because it interfaces with popular OSS(such as Linux). Open source advocates consider this to be both confusing and incorrect. OSI Certified is a trademark licensed only to people who are distributing software licensed under a license listed on the Open Source Initiative's list. Open source software and free software are different terms for software which comes with certain rights, or freedoms, for the user. They describe two approaches and philosophies towards free software. Open source and free software (or software libre) both describe software which is free from onerous licensing restrictions. It may be used, copied, studied, modified and redistributed without restriction. Free software is not the same as freeware, software available at zero price.

The definition of open source software was written to be almost identical to the free software definition. There are very few cases of software that is free software but is not open source

software, and vice versa. The difference in the terms is where they place the emphasis. Free software is defined in terms of giving the user freedom. This reflects the goal of the free software movement. Open source highlights that the source code is viewable to all and proponents of the term usually emphasize the quality of the software and how this is caused by the development models which are possible and popular among free and open source software projects. Free software licenses are not written exclusively by the FSF. The FSF and the OSI both list licenses which meet their respective definitions of free software. open source software and free software share an almost identical set of licenses. One exception is an early version of the Apple Public Source License, which was accepted by the OSI but rejected by the FSF because it did not allow private modified versions; this restriction was removed in later version of the license. There are now new versions that are approved by both the OSI and the FSF. The Open Source Initiative believes that more people will be convinced by the experience of freedom. The FSF believes that more people will be convinced by the concept of freedom. The FSF believes that knowledge of the concept is an essential requirement, insists on the use of the term free , and separates itself from the open source movement . The Open Source Initiative believes that free has three meanings: free as in beer, free as in freedom, and free as in unsellable. The problem with the term open source is it says nothing about the freedom to modify and redistribute, so it is used by people who think that source access without freedom is a sufficient definition. This possibility for misuse is the case for most of the licenses that make up Microsoft's shared source initiative.

12. Open source vs. source-available

Although the OSI definition of "open source software" is widely accepted, a small number of people and organizations use the term to refer to software where the source is available for viewing, but which may not legally be modified or redistributed. Such software is more often referred to as source-available, or as shared source, a term coined by Microsoft. Michael Tiemann, president of OSI, had criticized companies such as SugarCRM for promoting their software as "open source" when in fact it did not have an OSI-approved license. In SugarCRM's case, it was because the software is so-called "badgeware" since it specified a "badge" that must be displayed in the user interface (SugarCRM has since switched to GPLv3). Another example is Scilab, which calls itself "the open source platform for numerical computation but has a license that forbids commercial redistribution of modified versions. Because OSI does not have a registered trademark for the term "open source", its legal ability

to prevent such usage of the term is limited, but Tiemann advocates using public opinion from OSI, customers, and community members to pressure such organizations to change their license or to use a different term.

13. Development tools

In OSS development the participants, who are mostly volunteers, are distributed amongst different geographic regions so there is need for tools to aid participants to collaborate in source code development. Often these tools are also available as OSS. Revision control systems such as Concurrent Versions System (CVS) and later Subversion (svn) are examples of tools that help centrally manage the source code files and the changes to those files for a software project. Utilities that automate testing, compiling and bug reporting help preserve stability and support of software projects that have numerous developers but no managers, quality controller or technical support. Building systems that report compilation errors among different platforms include Tinderbox. Commonly used bugtrackers include Bugzilla and

14. GNATS.

Tools such as mailing lists, IRC, and instant messaging provide means of Internet communications between developers. The Web is also a core feature of all of the above systems. Some sites centralize all the features of these tools as a software development management system, including GNU Savannah, SourceForge, and BountySource.

15. Projects and organizations

Some of the more prominent organizations involved in OSS development include the Apache Software Foundation, creators of the Apache web server; a loose affiliation of developers headed by Linus Torvalds, creators of the Linux operating system kernel; the Eclipse Foundation, home of the Eclipse software development platform; the Debian Project, creators of the influential Debian GNU/Linux distribution; and the Mozilla Foundation, home of the Firefox web browser. Several Open Source programs have become defining entries in their space, including the GIMP image editing system; Sun's Java programming language and environment; the MySQL database system; the FreeBSD UNIXoperating system; Sun's 2 OpenOffice.org office productivity suite; and the Wireshark network packet sniffer and protocol analyser Open Source development is often performed "live and in public", using

services provided for free on the Internet, such as the Launchpad and SourceForge web sites, and using tools that are themselves Open Source, including the CVS and Subversion source control systems, and the GNU Compiler Collection.

16. Proprietary operating system

Several major proprietary operating systems eventually added recursive subdirectory capabilities also patterned after Multics. DEC's RSX-11M's "group, user" hierarchy evolved into VMS directories, CP/M's volumes evolved into MS-DOS 2.0+ subdirectories, and HP's MPE group.account hierarchy and IBM's SSP and OS/400 library systems were folded into broader POSIX file systems. Making the command interpreter an ordinary user-level program, with additional commands provided as separate programs, was another Multics innovation popularized by UNIX. The UNIXshell used the same language for interactive commands as for scripting (shell scripts there was no separate job control language like IBM's JCL). Since the shell and OS commands were "just another program", the user could choose (or even write) his own shell. New commands could be added without changing the shell itself. UNIXs innovative command-line syntax for creating chains of producer-consumer processes (pipelines) made a powerful programming paradigm (coroutines) widely available. Many later command-line interpreters have been inspired by the UNIXshell. A fundamental simplifying assumption of UNIXwas its focus on ASCII text for nearly all file formats. There were no "binary" editors in the original version of UNIX the entire system was configured using textual shell command scripts. The common denominator in the I/O system was the byte unlike "record-based" file systems.

The focus on text for representing nearly everything made UNIXpipes especially useful, and encouraged the development of simple, general tools that could be easily combined to perform more complicated ad hoc tasks. The focus on text and bytes made the system far more scalable and portable than other systems. Over time, text-based applications have also proven popular in application areas, such as printing languages (PostScript, ODF), and at the application layer of the Internet protocols, e.g., Telnet, FTP, SSH, SMTP, HTTP, SOAP and SIP. UNIXpopularized syntax for regular expressions that found widespread use. The UNIXprogramming interface became the basis for a widely implemented operating system interface standard (POSIX, see above). The C programming language soon spread beyond UNIX, and is now ubiquitous in systems and applications programming. Early UNIXdevelopers were important in bringing the concepts of modularity and reusability into

software engineering practice, spawning a "software tools" movement. UNIXprovided the TCP/IP networking protocol on relatively inexpensive computers, which contributed to the Internet explosion of worldwide real-time connectivity, and which formed the basis for implementations on many other platforms. This also exposed numerous security holes in the networking implementations. The UNIXpolicy of extensive on-line documentation and (for many years) ready access to all system source code raised programmer expectations, and contributed to the 1983 launch of the free software movement. Over time, the leading developers of UNIX (and programs that ran on it) established a set of cultural norms for developing software, norms which became as important and influential as the technology of UNIXitself; this has been termed the UNIXphilosophy.

Topic : Hardware

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Memory
- Understand the Hierarchy of storage
- Understand the Primary storage
- Understand the Secondary storage
- Understand the Tertiary storage
- Understand the Off-line storage
- Understand the processor
- Understand the Microcode

Definition/Overview:

Memory: Computer data storage, often called storage or memory, refers to computer components, devices, and recording media that retain digital data used for computing for some interval of time.

Hierarchy of storage: Various forms of storage, divided according to their distance from the central processing unit.

Primary storage: Primary storage, presently known as memory, is the only one directly accessible to the CPU. The CPU continuously reads instructions stored there and executes them as required. Any data actively operated on is also stored there in uniform manner.

Secondary storage: Secondary storage, or storage in popular usage, differs from primary storage in that it is not directly accessible by the CPU.

Tertiary storage: Large tape library. Tape cartridges placed on shelves in the front, robotic arm moving in the back.

Key Points:

1. Memory

Computer data storage, often called storage or memory, refers to computer components, devices, and recording media that retain digital data used for computing for some interval of time. Computer data storage provides one of the core functions of the modern computer, that of information retention. It is one of the fundamental components of all modern computers, and coupled with a central processing unit (CPU, a processor), implements the basic computer model used since the 1940s. In contemporary usage, memory usually refers to a form of semiconductor storage known as random access memory (RAM) and sometimes other forms of fast but temporary storage. Similarly, storage today more commonly refers to mass storage - optical discs, forms of magnetic storage like hard disks, and other types slower than RAM, but of a more permanent nature. Historically, memory and storage were respectively called primary storage and secondary storage. The contemporary distinctions are helpful, because they are also fundamental to the architecture of computers in general. As well, they reflect an important and significant technical difference between memory and mass storage devices, which has been blurred by the historical usage of the term storage. Nevertheless, this article uses the traditional nomenclature.

2. Hierarchy of storage

Various forms of storage, divided according to their distance from the central processing unit. The fundamental components of a general-purpose computer are arithmetic and logic unit, control circuitry, storage space, and input/output devices. Technology and capacity as in common home computers around 2005.

3. Primary storage

Primary storage, presently known as memory, is the only one directly accessible to the CPU. The CPU continuously reads instructions stored there and executes them as required. Any data actively operated on is also stored there in uniform manner. Historically, early computers used delay lines, Williamss tubes, or rotating magnetic drums as primary storage. By 1954, those unreliable methods were mostly replaced by magnetic core memory, which was still rather cumbersome. Undoubtedly, a revolution was started with the invention of a transistor that soon enabled then-unbelievable miniaturization of electronic memory via solid-state silicon chip technology. This led to a modern random access memory (RAM). It is small-sized, light, but quite expensive at the same time. (The particular types of RAM used for primary storage are also volatile, i.e. they lose the information when not powered).

Traditionally there are two more sub-layers of the primary storage, besides main large-capacity RAM:

- Processor registers are located inside the processor. Each register typically holds a word of data (often 32 or 64 bits). CPU instructions instruct the arithmetic and logic unit to perform various calculations or other operations on this data (or with the help of it). Registers are technically among the fastest of all forms of computer data storage.
- Processor cache is an intermediate stage between ultra-fast registers and much slower main memory. It's introduced solely to increase performance of the computer. Most actively used information in the main memory is just duplicated in the cache memory, which is faster, but of much lesser capacity. On the other hand it is much slower, but much larger than processor registers. Multi-level hierarchical cache setup is also commonly usedprimary cache being smallest, fastest and located inside the processor; secondary cache being somewhat larger and slower.

Main memory is directly or indirectly connected to the CPU via a memory bus, today sometimes referred to as a front side bus. It is actually comprised of two buses (not on the diagram): an address bus and a data bus. The CPU firstly sends a number through an address bus, a number called memory address that indicates the desired location of data. Then it reads or writes the data itself using the data bus. Additionally, a memory management unit (MMU) is a small device between CPU and RAM recalculating the actual memory address, for example to provide an abstraction of virtual memory or other tasks. As the RAM types used for primary storage are volatile (cleared at start up), a computer containing only such storage would not have a source to read instructions from, in order to start the computer. Hence, non-volatile primary storage containing a small startup program (BIOS) is used to bootstrap the computer, that is, to read a larger program from non-volatile secondary storage to RAM and start to execute it. A non-volatile technology used for this purpose is called ROM, for read-only memory (the terminology may be somewhat confusing as most ROM types are also capable of random access). Many types of "ROM" are not literally read only, as updates are possible; however it is slow and memory must be erased in large portions before it can be re-written. Some embedded systems run programs directly from ROM (or similar), because such programs are rarely changed. Standard computers do not store non-rudimentary programs in ROM; rather use large capacities of secondary storage, which is non-volatile as well, and not as costly.Recently, primary storage and secondary storage in some uses refer to what was historically called, respectively, secondary storage and tertiary storage.

4. Secondary storage

Secondary storage, or storage in popular usage, differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered downit is non-volatile. Per unit, it is typically also an order of magnitude less expensive than primary storage. Consequently, modern computer systems typically have an order of magnitude more secondary storage than primary storage and data is kept for a longer time there. In modern computers, hard disks are usually used as secondary storage. The time taken to access a given byte of information stored on a hard disk is typically a few thousandths of a second, or milliseconds. By contrast, the time taken to access a given byte of information stored in random access memory is measured in billionths of a second, or nanoseconds. This illustrates

the very significant access-time difference which distinguishes solid-state memory from rotating magnetic storage devices: hard disks are typically about a million times slower than memory. Rotating optical storage devices, such as CD and DVD drives, have even longer access times. Some other examples of secondary storage technologies are: flash memory (e.g. USB sticks or keys), floppy disks, magnetic tape, paper tape, punch cards, standalone RAM disks, and Zip drives.The secondary storage is often formatted according to a filesystem format, which provides the abstraction necessary to organize data into files and directories, providing also additional information (called metadata) describing the owner of a certain file, the access time, the access permissions, and other information. Most computer operating systems use the concept of virtual memory, allowing utilization of more primary storage capacity than is physically available in the system. As the primary memory fills up, the system moves the least-used chunks (pages) to secondary storage devices (to a swap file or page file), retrieving them later when they are needed. As more of these retrievals from slower secondary storage are necessary, the more the overall system performance is degraded.

5. Tertiary storage

Large tape library. Tape cartridges placed on shelves in the front, robotic arm moving in the back. Visible height of the library is about 180 cm. Tertiary storage or tertiary memory, provides a third level of storage. Typically it involves a robotic mechanism which will mount (insert) and dismount removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use. It is primarily used for archival of rarely accessed information since it is much slower than secondary storage (e.g. 5-60 seconds vs. 1-10 milliseconds). This is primarily useful for extraordinarily large data stores, accessed without human operators. Typical examples include tape libraries and optical jukeboxes. When a computer needs to read information from the tertiary storage, it will first consult a catalog database to determine which tape or disc contains the information. Next, the computer will instruct a robotic arm to fetch the medium and place it in a drive. When the computer has finished reading the information, the robotic arm will return the medium to its place in the library.

6. Off-line storage

Off-line storage, also known as disconnected storage, is computer data storage on a medium or a device that is not under the control of a processing unit.The medium is recorded, usually in a secondary or tertiary storage device, and then physically removed or disconnected. It must be inserted or connected by a human operator before a computer can access it again. Unlike tertiary storage, it cannot be accessed without human interaction. Off-line storage is used to transfer information, since the detached medium can be easily physically transported. Additionally in case a disaster, for example a fire, destroys the original data, a medium in a remote location will be probably unaffected, enabling disaster recovery. Off-line storage increases a general information security, since it is physically inaccessible from a computer, and data confidentiality or integrity cannot be affected by computer-based attack techniques. Also, if the information stored for archival purposes is accessed seldom or never, off-line storage is less expensive than tertiary storage. In modern personal computers, most secondary and tertiary storage media are also used for off-line storage. Optical discs and flash memory devices are most popular, and to much lesser extent removable hard disk drives. In enterprise uses, magnetic tape is predominant. Older examples are floppy disks, Zip disks, or punched cards.

7. The processor

The introduction of the microprocessor in the 1970s significantly affected the design and implementation of CPUs. Since the introduction of the first microprocessor (the Intel 4004) in 1970 and the first widely used microprocessor (the Intel 8080) in 1974, this class of CPUs has almost completely overtaken all other central processing unit implementation methods. Mainframe and minicomputer manufacturers of the time launched proprietary IC development programs to upgrade their older computer architectures, and eventually produced instruction set compatible microprocessors that were backward-compatible with their older hardware and software. Combined with the advent and eventual vast success of the now ubiquitous personal computer, the term "CPU" is now applied almost exclusively to microprocessors. Previous generations of CPUs were implemented as discrete components and numerous small integrated circuits (ICs) on one or more circuit boards. Microprocessors, on the other hand, are CPUs manufactured on a very small number of ICs; usually just one. The overall smaller CPU size as a result of being implemented on a single die means faster switching time because of physical factors like decreased gate parasitic capacitance. This has

allowed synchronous microprocessors to have clock rates ranging from tens of megahertz to several gigahertzes. Additionally, as the ability to construct exceedingly small transistors on an IC has increased, the complexity and number of transistors in a single CPU has increased dramatically. This widely observed trend is described by Moore's law, which has proven to be a fairly accurate predictor of the growth of CPU (and other IC) complexity to date. The integrated circuit from an Intel 8742, an 8-bit microcontroller that includes a CPU running at 12 MHz, 128 bytes of RAM, 2048 bytes of EPROM, and I/O in the same chip. While the complexity, size, construction, and general form of CPUs have changed drastically over the past sixty years, it is notable that the basic design and function has not changed much at all. Almost all common CPUs today can be very accurately described as von Neumann stored-program machines. As the aforementioned Moore's law continues to hold true, concerns have arisen about the limits of integrated circuit transistor technology. Extreme miniaturization of electronic gates is causing the effects of phenomena like electromigration and subthreshold leakage to become much more significant. These newer concerns are among the many factors causing researchers to investigate new methods of computing such as the quantum computer, as well as to expand the usage of parallelism and other methods that extend the usefulness of the classical von Neumann model.

8. Microcode

Microcode is a layer of lowest-level instructions involved in the implementation of machine code instructions in many computers and other processors; it resides in a special high-speed memory and translates machine instructions into sequences of detailed circuit-level operations. It helps separate the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also makes it feasible to build complex multi-step instructions while still reducing the complexity of the electronic circuitry compared to other methods. Writing microcode is called microprogramming and the microcode for a given processor is often called a microprogram. The microcode is normally written by the CPU engineer during the design phase. It is generally not meant to be visible or changeable by a normal programmer, or even an assembly programmer. Unlike machine code which often retains backwards compatibility, microcode only runs on the exact CPU model for which it's designed. Microcode can be used to let one microarchitecture emulate another, usually more powerful, architecture. Some hardware vendors, especially IBM, also use the term microcode as a synonym for firmware, whether or not it actually implements the

microprogramming of a processor. Even simple firmware, such as the one used in a hard drive, is sometimes described as microcode. Such use is not discussed here. The elements composing a microprogram exist on a lower conceptual level than a normal application program. Each element is differentiated by the "micro" prefix to avoid confusion: microinstruction, microassembler, microprogrammer, microarchitecture, etc. The microcode usually does not reside in the main memory, but in a special high speed memory, called the control store. It might be either read-only or read-write memory. In the latter case the microcode would be loaded into the control store from some other storage medium as part of the initialization of the CPU, and it could be altered to correct bugs in the instruction set, or to implement new machine instructions.

Microprograms consist of series of microinstructions. These microinstructions control the CPU at a very fundamental level of hardware circuitry. For example, a single typical microinstruction might specify the following operations:

- Connect Register 1 to the "A" side of the ALU
- Connect Register 7 to the "B" side of the ALU
- Set the ALU to perform two's-complement addition
- Set the ALU's carry input to zero
- Store the result value in Register 8
- Update the "condition codes" with the ALU status flags ("Negative", "Zero", "Overflow", and "Carry")
- Microjump to MicroPC nnn for the next microinstruction

To simultaneously control all processor's features in one cycle, the microinstruction is often as wide as 50 or more bits. Microprograms are carefully designed and optimized for the fastest possible execution, since a slow microprogram would yield a slow machine instruction which would in turn cause all programs using that instruction to be slow.

9. The reason for microprogramming

Microcode was originally developed as a simpler method of developing the control logic for a computer. Initially CPU instruction sets were "hard wired". Each step needed to fetch, decode and execute the machine instructions (including any operand address calculations, reads and writes) were controlled directly by combinatorial logic and rather minimal sequential state

machine circuitry. While very efficient, the need for powerful instruction sets with multi-step addressing and complex operations (see below) made such "hard-wired" processors difficult to design and debug; highly encoded and varied-length instructions can contribute to this as well, especially when very irregular encodings are used. Microcode simplified the job by allowing much of the processor's behavior and programming model be defined via microprogram routines rather than by dedicated circuitry. Even late in the design process, microcode could easily be changed, whereas hard wired CPU designs were very cumbersome to change, so this greatly facilitated CPU design. In the 1940s through the late 1970s, much programming was done in assembly language; higher level instructions meant greater programmer productivity, so an important advantage of microcode was the relative ease by which powerful machine instructions could be defined. During the 1970s, CPU speeds grew more quickly than memory speeds and numerous techniques such as memory block transfer, memory pre-fetch and multi-level caches were used to alleviate this. High level machine instructions, made possible by microcode, helped further, as fewer more complex machine instructions require less memory bandwidth. For example, an operation on a character string could be done as a single machine instruction, thus avoiding multiple instruction fetches. Architectures with instruction sets implemented by complex microprograms included the IBM System/360 and Digital Equipment Corporation VAX. The approach of increasingly complex microcode-implemented instruction sets was later called CISC. A middle way, used in many microprocessors, is to use PLAs and/or ROMs (instead of combinatorial logic) mainly for instruction decoding, and let a simple state machine (without much, or any, microcode) do most of the sequencing. The various practical uses of microcode and related techniques (such as PLAs) have been numerous over the years, as well as approaches to where, and to which extent, it should be used. It is still used in modern CPU designs. A processor's microprograms operate on a more primitive, totally different and much more hardware-oriented architecture than the assembly instructions visible to normal programmers. In coordination with the hardware, the microcode implements the programmer-visible architecture. The underlying hardware need not have a fixed relationship to the visible architecture. This makes it possible to implement a given instruction set architecture on a wide variety of underlying hardware micro-architectures. Doing so is important if binary program compatibility is a priority. Those way previously existing programs can run on totally new hardware without requiring revision and recompilation. However there may be a performance penalty for this approach. The tradeoffs between application backward compatibility vs CPU performance are hotly debated by CPU design engineers. The IBM

System/360 has a 32-bit architecture with 16 general-purpose registers, but most of the System/360 implementations actually use hardware that implemented a much simpler underlying microarchitecture; for example, the System/360 Model 30 had 8-bit data paths to the arithmetic logic unit (ALU) and main memory and implemented the general-purpose registers in a special unit of higher-speed core memory, and the System/360 Model 40 had 8-bit data paths to the ALU and 16-bit data paths to main memory and also implemented the general-purpose registers in a special unit of higher-speed core memory. The Model 50 and Model 65 had full 32-bit data paths and implemented the general-purpose registers in faster transistor circuits. In this way, microprogramming enabled IBM to design many System/360 models with substantially different hardware and spanning a wide range of cost and performance, while making them all architecturally compatible. This dramatically reduced the amount of unique system software that had to be written for each model.

A similar approach was used by Digital Equipment Corporation in their VAX family of computers. Initially a 32-bit TTL processor in conjunction with supporting microcode implemented the programmer-visible architecture. Later VAX versions used different microarchitectures, yet the programmer-visible architecture didn't change.

Microprogramming also reduced the cost of field changes to correct defects (bugs) in the processor; a bug could often be fixed by replacing a portion of the microprogram rather than by changes being made to hardware logic and wiring.

10. Input/output devices

In computing, input/output, or I/O, refers to the communication between an information processing system (such as a computer), and the outside world possibly a human, or another information processing system. Inputs are the signals or data received by the system, and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a person (or other system) to communicate with a computer. For instance, keyboards and mouses are considered input devices of a computer, while monitors and printers are considered output devices of a computer. Devices for communication between computers, such as modems and network cards, typically serve for both input and output. Note that the designation of a device as either input or output depends on the perspective. Mouses and keyboards take as input physical movement that the human user outputs and convert it into signals that a computer

can understand. The output from these devices is input for the computer. Similarly, printers and monitors take as input signals that a computer outputs. They then convert these signals into representations that human users can see or read. (For a human user the process of reading or seeing these representations is receiving input.) In computer architecture, the combination of the CPU and main memory (i.e. memory that the CPU can read and write to directly, with individual instructions) is considered the brain of a computer, and from that point of view any transfer of information from or to that combination, for example to or from a disk drive, is considered I/O. The CPU and its supporting circuitry provide memory-mapped I/O that is used in low-level computer programming in the implementation of device drivers. Higher-level operating system and programming facilities employ separate, more abstract I/O concepts and primitives. For example, most operating systems provide application programs with the concept of files. The C and C++ programming languages, and operating systems in the UNIXfamily, traditionally abstract files and devices as streams, which can be read or written, or sometimes both. The C standard library provides functions for manipulating streams for input and output. An alternative to special primitive functions is the I/O monad, which permits programs to just describe I/O, and the actions are carried out outside the program. This is notable because the I/O functions would introduce side-effects to any programming language, but now purely functional programming is practical.

10. I/O Interface

I/O Interface is required whenever the I/O device is driven by the processor. The interface must have necessory logic to interpret the device address generated by the processor. Handshaking should be implemented by the interface using appropriate commands like (BUSY, READY, WAIT) the processor can communicate with I/O device through the interface. If incase different data formated being exchanged; interface must be able to convert serial data to parallel form and vice-versa. There must be provision for generating interrupts and the corresponding type numbers for further processing by the processor if requiredA computer that uses memory-mapped I/O accesses hardware by reading and writing to specific memory locations, using the same assembler language instructions that computer would normally use to access memory.

11. Secondary storage

Secondary storage, or storage in popular usage, differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered downit is non-volatile. Per unit, it is typically also an order of magnitude less expensive than primary storage. Consequently, modern computer systems typically have an order of magnitude more secondary storage than primary storage and data is kept for a longer time there. In modern computers, hard disks are usually used as secondary storage. The time taken to access a given byte of information stored on a hard disk is typically a few thousandths of a second, or milliseconds. By contrast, the time taken to access a given byte of information stored in random access memory is measured in billionths of a second, or nanoseconds. This illustrates the very significant access-time difference which distinguishes solid-state memory from rotating magnetic storage devices: hard disks are typically about a million times slower than memory. Rotating optical storage devices, such as CD and DVD drives, have even longer access times. Some other examples of secondary storage technologies are: flash memory (e.g. USB sticks or keys), floppy disks, magnetic tape, paper tape, punch cards, standalone RAM disks, and Zip drives. The secondary storage is often formatted according to a filesystem format, which provides the abstraction necessary to organize data into files and directories, providing also additional information (called metadata) describing the owner of a certain file, the access time, the access permissions, and other information. Most computer operating systems use the concept of virtual memory, allowing utilization of more primary storage capacity than is physically available in the system. As the primary memory fills up, the system moves the least-used chunks (pages) to secondary storage devices (to a swap file or page file), retrieving them later when they are needed. As more of these retrievals from slower secondary storage are necessary, the more the overall system performance is degraded.

12. Tertiary storage

Tertiary storage or tertiary memory provides a third level of storage. Typically it involves a robotic mechanism which will mount (insert) and dismount removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use. It is primarily used for archival of rarely accessed information

since it is much slower than secondary storage (e.g. 5-60 seconds vs. 1-10 milliseconds). This is primarily useful for extraordinarily large data stores, accessed without human operators. Typical examples include tape libraries and optical jukeboxes. When a computer needs to read information from the tertiary storage, it will first consult a catalog database to determine which tape or disc contains the information. Next, the computer will instruct a robotic arm to fetch the medium and place it in a drive. When the computer has finished reading the information, the robotic arm will return the medium to its place in the library.

13. Off-line storage

Off-line storage, also known as disconnected storage, is a computer data storage on a medium or a device that is not under the control of a processing unit.The medium is recorded, usually in a secondary or tertiary storage device, and then physically removed or disconnected. It must be inserted or connected by a human operator before a computer can access it again. Unlike tertiary storage, it cannot be accessed without human interaction. Off-line storage is used to transfer information, since the detached medium can be easily physically transported. Additionally in case a disaster, for example a fire, destroys the original data, a medium in a remote location will be probably unaffected, enabling disaster recovery. Off-line storage increases a general information security, since it is physically inaccessible from a computer, and data confidentiality or integrity cannot be affected by computer-based attack techniques. Also, if the information stored for archival purposes is accessed seldom or never, off-line storage is less expensive than tertiary storage. In modern personal computers, most secondary and tertiary storage media are also used for off-line storage. Optical discs and flash memory devices are most popular, and to much lesser extent removable hard disk drives. In enterprise uses, magnetic tape is predominant. Older examples are floppy disks, Zip disks, or punched cards.

14. Communication hardware

All networks are made up of basic hardware building blocks to interconnect network nodes, such as Network Interface Cards (NICs), Bridges, Hubs, Switches, and Routers. In addition, some method of connecting these building blocks is required, usually in the form of galvanic cable (most commonly Category 5 cable). Less common are microwave links (as in IEEE 802.12) or optical cable ("optical fiber"). An Ethernet card may also be required.

15. Network interface cards

A network card, network adapter or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a computer network. It provides physical access to a networking medium and often provides a low-level addressing system through the use of MAC addresses. It allows users to connect to each other either by using cables or wirelessly. The NIC provides the transfer of data in megabytes.

16. Repeaters

A repeater is an electronic device that receives a signal and retransmits it at a higher power level, or to the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable runs longer than 100 meters away from the computer.

17. Hubs

A hub contains multiple ports. When a packet arrives at one port, it is copied unmodified to all ports of the hub for transmission. The destination address in the frame is not changed to a broadcast address.

18. Bridges

A network bridge connects multiple network segments at the data link layer (layer 2) of the OSI model. Bridges do not promiscuously copy traffic to all ports, as hubs do, but learn which MAC addresses are reachable through specific ports. Once the bridge associates a port and an address, it will send traffic for that address only to that port. Bridges do send broadcasts to all ports except the one on which the broadcast was received. Bridges learn the association of ports and addresses by examining the source address of frames that it sees on various ports. Once a frame arrives through a port, its source address is stored and the bridge assumes that MAC address is associated with that port. The first time that a previously unknown destination address is seen, the bridge will forward the frame to all ports other than the one on which the frame arrived.

19. Linking the component

The following list presents categories used for classifying networks.

Connection method- Computer networks can also be classified according to the hardware and software technology that is used to interconnect the individual devices in the network, such as Optical fiber, Ethernet, Wireless LAN, HomePNA, or Power line communication.

Ethernet uses physical wiring to connect devices. Frequently deployed devices include hubs, switches, bridges and/or routers.

Wireless LAN technology is designed to connect devices without wiring. These devices use radio waves or infrared signals as a transmission medium.

Scale- Based on their scale, networks can be classified as Local Area Network (LAN), Wide Area Network (WAN), Metropolitan Area Network (MAN), Personal Area Network (PAN), Virtual Private Network (VPN), Campus Area Network (CAN), Storage Area Network (SAN), etc.

20. Functional relationship (network architecture)

Computer networks may be classified according to the functional relationships which exist among the elements of the network, e.g., Active Networking, Client-server and Peer-to-peer (workgroup) architecture.

21. Network topology

Computer networks may be classified according to the network topology upon which the network is based, such as bus network, star network, ring network, mesh network, star-bus network, tree or hierarchical topology network. Network topology signifies the way in which devices in the network see their logical relations to one another. The use of the term "logical" here is significant. That is, network topology is independent of the "physical" layout of the network. Even if networked computers are physically placed in a linear arrangement, if they are connected via a hub, the network has a Star topology, rather than a bus topology. In this regard the visual and operational characteristics of a network are distinct; the logical network

topology is not necessarily the same as the physical layout. Networks may be classified based on the method of data used to convey the data; these include digital and analog networks.

Topic : Software And Data

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Computer hardware
- Understand the Motherboard
- Understand the Current
- Understand the Power supply
- Understand the Video display controller
- Understand the Removable media devices
- Understand the Computer software

Definition/Overview:

Motherboard: Motherboard - It is the "body" or mainframe of the computer, through which all other components interface.

Video display controller: Produces the output for the visual display unit.

Computer software: Computer software or just software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system.

Relationship to computer hardware: Computer software is so called to distinguish it from computer hardware, which encompasses the physical interconnections and devices required to store and execute (or run) the software. At the lowest level, software consists of a machine language specific to an individual processor.

Key Points:

1. Computer hardware

Though a pc comes in many different form factors, a typical personal computer consists of a case or chassis in a tower shape (desktop) and the following parts:

2. Motherboard

Motherboard - It is the "body" or mainframe of the computer, through which all other components interface. Central processing unit (CPU) - Performs most of the calculations which enable a computer to function, sometimes referred to as the "backbone" or "brain" of the computer. Computer fan - Used to lower the temperature of the computer; a fan is almost always attached to the CPU, and to the back of the case, also known as the 'muffin fan. Firmware is system specific read only memory.

Internal Buses - Connections to various internal components.

3. Current

- PCI (being phased out for graphic cards but still used for other uses)
- PCI Express (PCI-E)
- USB
- FireWire

- HyperTransport
- Intel QuickPath (expected in 2008)
- Obsolete
- AGP (Obsolete graphic card bus)
- ISA (obsolete in PCs, but still used in industrial computers)
- VLB VESA Local Bus (outdated)
- External Bus Controllers - used to connect to external peripherals, such as printers and input devices. These ports may also be based upon expansion cards, attached to the internal

4. Power supply

A case control, and (usually) a cooling fan, and supplies power to run the rest of the computer, the most common types of power supplies are mechanic shed (old) but the standard for PCs actually are ATX and Micro ATX. Power supply is the heart of our computer.

5. Video display controller

It produces the output for the visual display unit. This will either be built into the motherboard or attached in its own separate slot (PCI, PCI-E, PCI-E 2.0, or AGP), in the form of a graphics card.

6. Removable media devices

- CD (compact disc) - the most common type of removable media, inexpensive but has a short life-span.
- CD-ROM Drive - a device used for reading data from a CD.
- CD Writer - a device used for both reading and writing data to and from a CD.
- DVD (digital versatile disc) - a popular type of removable media that is the same dimensions as a CD but stores up to 6 times as much information. It is the most common way of transferring digital video.
- DVD-ROM Drive - a device used for reading data from a DVD.
- DVD Writer - a device used for both reading and writing data to and from a DVD.
- DVD-RAM Drive - a device used for rapid writing and reading of data from a special type of DVD.
- Blu-ray - a high-density optical disc format for the storage of digital information, including high-definition video.
- BD-ROM Drive - a device used for reading data from a Blu-ray disc.
- BD Writer - a device used for both reading and writing data to and from a Blu-ray disc.
- HD DVD - a high-density optical disc format and successor to the standard DVD. It was a discontinued competitor to the Blu-ray format.
- Floppy disk - an outdated storage device consisting of a thin disk of a flexible magnetic storage medium. Used today mainly for loading RAID drivers.

- Zip drive - an outdated medium-capacity removable disk storage system, first introduced by Iomega in 1994.
- USB flash drive - a flash memory data storage device integrated with a USB interface, typically small, lightweight, removable, and rewritable.
- Tape drive - a device that reads and writes data on a magnetic tape, used for long term storage.

7. Computer software

Computer software, or just software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system.

Application software such as word processors which perform productive tasks for users. Firmware which is software programmed resident to electrically programmable memory devices on board mainboards or other types of integrated hardware carriers. Middleware which controls and co-ordinates distributed systems. System software such as operating systems, which interface with hardware to provide the necessary services for application software. Software Testing is a domain independent of development and programming, It consists various methods to test and declare a software product fit before it can be launched for use by either an indivudual or a group. Many tests on functionality, performance and appearance are conducted by modern testers with various tools such as QTP, Load runner, Black box testing etc to edit a checklist of requirements against the developed code. ISTQB is a certification that is in demand for engineers who want to pursue a career in testing. Testware which is an umbrella term or container term for all utilities and application software that serve in combination for testing asoftware package but not necessarily may optionally contribute to operational purposes. As such, testware is not a standing configuration but merely a working environment for application software or subsets thereof. Software includes websites, programs, video games etc. that are coded by programming languages like C, C++, etc. "Software" is sometimes used in a broader context to mean anything which is not hardware but which is used with hardware, such as film, tapes and records.

8. Relationship to computer hardware

Computer software is so called to distinguish it from computer hardware, which encompasses the physical interconnections and devices required to store and execute (or run) the software. At the lowest level, software consists of a machine language specific to an individual processor. A machine language consists of groups of binary values signifying processor instructions which change the state of the computer from its preceding state. Software is an ordered sequence of instructions for changing the state of the computer hardware in a particular sequence. It is usually written in high-level programming languages that are easier and more efficient for humans to use (closer to natural language) than machine language. High-level languages are compiled or interpreted into machine language object code. Software may also be written in an assembly language, essentially, a mnemonic representation of a machine language using a natural language alphabet. Assembly language must be assembled into object code via an assembler. The term "software" was first used in this sense by John W. Tukey in 1958. In computer science and software engineering, computer software is all computer programs. The theory that is the basis for most modern software was first proposed by Alan Turing in his 1935 essay.

9. Types of software

Practical computer systems divide software systems into three major classes: system software, programming software and application software, although the distinction is arbitrary, and often blurred.

- System software
- System software helps run the computer hardware and computer system. It includes:
- device drivers,
- operating systems,
- servers,
- utilities,
- windowing systems,

10. Software Architecture

Users often see things differently than programmers. People who use modern general purpose computers (as opposed to embedded systems, analog computers, supercomputers, etc.) usually see three layers of software performing a variety of tasks: platform, application, and user software. Platform software: Platform includes the firmware, device drivers, an operating system, and typically a graphical user interface which, in total, allow a user to interact with the computer and its peripherals (associated equipment). Platform software often comes bundled with the computer. On a PC you will usually have the ability to change the platform software. Application software: Application software or Applications are what most people think of when they think of software. Typical examples include office suites and video games. Application software is often purchased separately from computer hardware. Sometimes applications are bundled with the computer, but that does not change the fact that they run as independent applications. Applications are almost always independent programs from the operating system, though they are often tailored for specific platforms. Most users think of compilers, databases, and other "system software" as applications. User-written software: End-user development tailors systems to meet users' specific needs. User software include spreadsheet templates, word processor macros, scientific simulations, and scripts for graphics and animations. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is. Depending on how competently the user-written software has been integrated into default application packages, many users may not be aware of the distinction between the original packages, and what has been added by co-workers.

11. Documentation

Most software has software documentation so that the end user can understand the program, what it does and how to use it. Without a clear documentation software can be hard to use and especially if it is very specialized and relatively complex software like the Photoshop, AutoCAD, etc. Developer documentation may also exist, either with the code as comments and/or as separate files, detailing how the programs works and can be modified.

12. Library

A executable is almost always not sufficiently complete for direct execution. Software libraries include collections of functions and functionality that may be embedded in other applications. Operating systems include many standard Software libraries, and applications are often distributed with their own libraries.

13. Execution

Computer software has to be "loaded" into the computer's storage (such as a hard drive, memory, or RAM). Once the software has loaded, the computer is able to execute the software. This involves passing instructions from the application software, through the system software, to the hardware which ultimately receives the instruction as machine code. Each instruction causes the computer to carry out an operation -- moving data, carrying out a computation, or altering the control flow of instructions. Data movement is typically from one place in memory to another. Sometimes it involves moving data between memory and registers which enable high-speed data access in the CPU. Moving data, especially large amounts of it, can be costly. So, this is sometimes avoided by using "pointers" to data instead. Computations include simple operations such as incrementing the value of a variable data element. More complex computations may involve many operations and data elements together.

14. Quality and reliability

Software quality, Software testing, and Software reliability Software quality is very important, especially for commercial and system software like Microsoft Office, Microsoft Windows, Linux, etc. If software is faulty (buggy), it can delete a person's work, crash the computer and do other unexpected things. Faults and errors are called "bugs". Many bugs are discovered and eliminated (debugged) through software testing. However, software testing rarely -- if ever -- eliminates every bug; some programmers say that "every program has at least one more bug" (Lubarsky's Law). All major software companies, such as Microsoft, Novell and Sun Microsystems, have their own software testing departments with the specific goal of just testing. Software can be tested through unit testing, regression testing and other methods, which are done manually, or most commonly, automatically, since the amount of code to be tested can be quite large. For instance, NASA has extremely rigorous software

testing procedures for its Space Shuttle and other programs because faulty software can crash the whole program and make the vehicle not functional, at great expense.

15. License

The software's license gives the user the right to use the software in the licensed environment. Some software comes with the license when purchased off the shelf, or an OEM license when bundled with hardware. Other software comes with a free software license, granting the recipient the rights to modify and redistribute the software. Software can also be in the form of freeware or shareware.

16. Patents

Software can be patented; however software patents can be controversial in the software industry with many people holding different views about it. Some believe that they hinder software development, while others argue that software patents provide an important incentive to spur software innovation. The controversy over software patents is that a specific algorithm or technique that the software has cannot be duplicated by others and is considered an intellectual property and copyright infringement depending on the severity.

17. Data

Data refer to a collection of facts usually collected as the result of experience, observation or experiment, or processes within a computer system, or a set of premises. This may consist of numbers, words, or images, particularly as measurements or observations of a set of variables. Data are often viewed as a lowest level of abstraction from which information and knowledge are derived.In computer science, data is anything in a form suitable for use with a computer. Data is often distinguished from programs. A program is a set of instructions that detail a task for the computer to perform. In this sense, data is thus everything that is not program code. In an alternate usage, binary files (which are not human-readable) are sometimes called "data" as distinguished from human-readable "text". The total amount of digital data in 2007 was estimated to be 281 billion gigabytes.

18. Data vs. programs

Fundamentally, computers follow the instructions they are given. A set of instructions to perform a given task (or tasks) is called a "program". In the nominal case, the program, as executed by the computer, will consist of binary machine code. The elements of storage manipulated by the program, but not actually executed by the CPU, contain data. Typically, different files are used to store programs vs. data. Executable files contain programs; all other files are data files. However, executable files may also contain data which is "built-in" to the program. In particular, some executable files have a data segment, which nominally contains constants and initial values (both data). For example: A user might first instruct the operating system to load a word processor program from one file, and then edit a document stored in another file. In this example, the document would be considered data. If the word processor also features a spell checker, then the dictionary (word list) for the spell checker would also be considered data. The algorithms used by the spell checker to suggest corrections would be considered code. The line between program and data can become blurry. An interpreter, for example, is a program. The input data to an interpreter is itself a programjust not one expressed in native machine language. In many cases, the interpreted program will be a human-readable text file, which is manipulated with a text editormore normally associated with plain text data.

Topic : Linking The Hardware Components

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Computer architecture
- Understand the Sub-definitions
- Understand the Design goals
- Understand the Performance
- Understand the Hardware/ Software interface
- Understand the Interfaces in practice

- Understand the GUI

Definition/Overview:

Computer architecture: Computer architecture in computer engineering is the conceptual design and fundamental operational structure of a computer system.
Design goals: The exact form of a computer system depends on the constraints and goals for which it was optimized.
Performance: Computer performance is often described in terms of clock speed (usually in MHz or GHz).

Hardware/ Software interface: Interface generally refers to an abstraction that an entity provides of itself to the outside.

Interfaces in practice: A piece of 'software' provides access to computer resources (such as memory, CPU, storage, etc.) by its underlying computer system; the availability of these resources to other software can have major ramificationssometimes disastrous onesfor its functionality and stability.

Key Points:

1. Computer architecture
Computer architecture in computer engineering is the conceptual design and fundamental operational structure of a computer system. It is a blueprint and functional description of requirements and design implementations for the various parts of a computer, focusing largely on the way by which the central processing unit (CPU) performs internally and accesses addresses in memory.
It may also be defined as the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.
2. Sub-definitions
Some practitioners of computer architecture at companies such as Intel and AMD use more fine distinctions:

- Macroarchitecture - architectural layers that are more abstract than microarchitecture, e.g. ISA
- ISA (Instruction Set Architecture) - as defined above

- Assembly ISA - a smart assembler may convert an abstract assembly language common to a group of machines into slightly different machine language for different implementations
- Programmer Visible Macroarchitecture - higher level language tools such as compilers may define a consistent interface or contract to programmers using them, abstracting differences between underlying ISA, UISA, and microarchitectures. E.g. the C, C++, or Java standards define different Programmer Visible Macroarchitecture - although in practice the C microarchitecture for a particular computer includes
- UISA (Microcode Instruction Set Architecture) - a family of machines with different hardware level microarchitectures may share a common microcode architecture, and hence a UISA.
- Pin Architecture - the set of functions that a microprocessor is expected to provide, from the point of view of a hardware platform. E.g. the x86 A20M, FERR/IGNNE or FLUSH pins, and the messages that the processor is expected to emit after completing a cache invalidation so that external caches can be invalidated. Pin architecture functions are more flexible than ISA functions - external hardware can adapt to changing encodings, or changing from a pin to a message - but the functions are expected to be provided in successive implementations even if the manner of encoding them changes.

3. Design goals

The exact form of a computer system depends on the constraints and goals for which it was optimized. Computer architectures usually trade off standards, cost, memory capacity, latency and throughput. Sometimes other considerations, such as features, size, weight, reliability, expandability and power consumption are factors as well.

The most common scheme carefully chooses the bottleneck that most reduces the computer's speed. Ideally, the cost is allocated proportionally to assure that the data rate is nearly the same for all parts of the computer, with the most costly part being the slowest. This is how skillful commercial integrators optimize personal computers.

4. Performance

Computer performance is often described in terms of clock speed (usually in MHz or GHz). This refers to the cycles per second of the main clock of the CPU. However, this metric is somewhat misleading, as a machine with a higher clock rate may not necessarily have higher performance. As a result manufacturers have moved away from clock speed as a measure of performance. Computer performance can also be measured with the amount of cache a processor has. If the speed, MHz or GHz, were to be a car then the cache is like a traffic light.

No matter how fast the car goes, it still will be stopped by a red traffic light. The higher the speed, and the greater the cache, the faster a processor runs. Modern CPUs can execute multiple instructions per clock cycle, which dramatically speeds up a program. Other factors influence speed, such as the mix of functional units, bus speeds, available memory, and the type and order of instructions in the programs being run. There are two main types of speed, latency and throughput. Latency is the time between the start of a process and its completion. Throughput is the amount of work done per unit time. Interrupt latency is the guaranteed maximum response time of the system to an electronic event (e.g. when the disk drive finishes moving some data). Performance is affected by a very wide range of design choices for example, pipelining a processor usually makes latency worse (slower) but makes throughput better. Computers that control machinery usually need low interrupt latencies. These computers operate in a real-time environment and fail if an operation is not completed in a specified amount of time. For example, computer-controlled anti-lock brakes must begin braking almost immediately after they have been instructed to brake. The performance of a computer can be measured using other metrics, depending upon its application domain. A system may be CPU bound (as in numerical calculation), I/O bound (as in a webserving application) or memory bound (as in video editing). Power consumption has become important in servers and portable devices like laptops. Benchmarking tries to take all these factors into account by measuring the time a computer takes to run through a series of test programs. Although benchmarking shows strengths, it may not help one to choose a computer. Often the measured machines split on different measures. For example, one system might handle scientific applications quickly, while another might play popular video games more smoothly. Furthermore, designers have been known to add special features to their products, whether in hardware or software, which permit a specific benchmark to execute quickly but which do not offer similar advantages to other, more general tasks.

5. Hardware/ Software interface

Interface generally refers to an abstraction that an entity provides of itself to the outside. This separates the methods of external communication from internal operation, and allows it to be internally modified without affecting the way outside entities interact with it, as well as provide multiple abstractions of it. It may also provide a means of translation between entities which do not speak the same language, such as between a human and a computer. Because

interfaces are a form of indirection, some additional overhead is incurred versus direct communication.

The interface between a human and a computer is called a user interface. Interfaces between hardware components are physical interfaces. This article deals with software interfaces, which exist between separate software components and provide a programmatic mechanism by which these components can communicate.

6. Interfaces in practice

A piece of 'software' provides access to computer resources (such as memory, CPU, storage, etc.) by its underlying computer system; the availability of these resources to other software can have major ramificationssometimes disastrous onesfor its functionality and stability. A key principle of design is to prohibit access to all resources by default, allowing access only through well-defined entry points, i.e. interfaces. The types of access that interfaces provide between software components can include: constants, data types, types of procedures, exception specifications and method signatures. In some instances, it may be useful to define variables as part of the interface. It often also specifies the functionality of those procedures and methods, either by comments or (in some experimental languages) by formal logical assertions. The interface of a software module A is deliberately kept separate from the implementation of that module. The latter contains the actual code of the procedures and methods described in the interface, as well as other "private" variables, procedures, etc.. Any other software module B (which can be referred to as a client to A) that interacts with A is forced to do so only through the interface. One practical advantage of this arrangement is that replacing the implementation of A by another one that meets the same specifications of the interface should not cause Bto failas long as its use of A complies with the specifications of the interface

7. GUI

A graphical user interface (GUI, IPA: /ˈguːi/) is a type of user interface which allows people to interact with electronic devices such as computers; hand-held devices such as MP3 Players, Portable Media Players or Gaming devices; household appliances and office equipment. A GUI offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

The term GUI is historically restricted to the scope of two-dimensional display screens with display resolutions capable of describing generic information, in the tradition of the computer science research at Palo Alto Research Center (PARC). The term GUI earlier might have been applicable to other high-resolution types of interfaces that are non-generic, such as videogames, or not restricted to flat screens, like volumetric displays.

8. Precursors

The precursor to GUIs was invented by researchers at the Stanford Research Institute, led by Douglas Engelbart. They developed the use of text-based hyperlinks manipulated with a mouse for the On-Line System. The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC, who went beyond text-based hyperlinks and used a GUI as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. As a result, some people call this class of interface a PARC User Interface (PUI) (note that PUI is also an acronym for perceptual user interface). Ivan Sutherland developed a pointer-based system called Sketchpad in 1963. It used a light-pen to guide the creation and manipulation of objects in engineering drawings.

9. PARC User Interface

The PARC User Interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC User Interface employs a pointing device in addition to a keyboard. These aspects can be emphasized by using the alternative acronym WIMP, which stands for Windows, Icons, Menus and Pointing device.

10. Evolution

Following PARC the first GUI-centric computer operating model was the Xerox 8010 Star Information System in 1981 followed by the Apple Lisa (which presented concept of menu bar as well as window controls), in 1982 and the Atari ST and Commodore Amiga in 1985. The GUIs familiar to most people today are Microsoft Windows, Mac OS X, and the X Window System interfaces. Apple, IBM and Microsoft used many of Xerox's ideas to develop products, and IBMs Common User Access specifications formed the basis of the user interface found in Microsoft Windows, IBM OS/2 Presentation Manager, and the UNIXMotif toolkit and window manager. These ideas evolved to create the interface found in current versions of Microsoft Windows, as well as in Mac OS X and various desktop environments for Unix-like operating systems, such as Linux. Thus most current GUIs have largely common idioms.

11. Components

A GUI uses a combination of technologies and devices to provide a platform the user can interact with, for the tasks of gathering and producing information. The most common combination in GUIs is the WIMP paradigm, especially in personal computers. Further information: WIMP (computing). This style of interaction uses a physical input device to control the position of a cursor and presents information organized in windows and represented with icons. Available commands are compiled together in menus and actioned through the pointing device. A window manager facilitates the interactions between windows, applications, and the windowing system. The windowing system handles hardware devices such as pointing devices and graphics hardware, as well as the positioning of the cursor. Further information: Window manager In personal computers all these elements are modeled through a desktop metaphor, to produce a simulation called a desktop environment in which the display represents a desktop, upon which documents and folders of documents can be placed. Window managers and other software combine to simulate the desktop environment with varying degrees of realism. Further information: Desktop environment

12. Post-WIMP Interfaces

Smaller mobile devices such as PDAs and smartphones typically use the WIMP elements with different unifying metaphors, due to constraints in space and available input devices. Applications for which WIMP is not well suited may use newer interaction techniques, collectively named as post-WIMP user interfaces. Some touch-screen-based devices such as Apple's iPhone currently use post-WIMP styles of interaction. The iPhone's use of more than one finger in contact with the screen allows actions such as pinching and rotating, which are not supported by a single pointer and mouse. A class of GUIs sometimes referred to as post-WIMP include 3D compositing window manager such as Compiz, Desktop Window Manager, and LG3D. Some post-WIMP interfaces may be better suited for applications which model immersive 3D environments, such as Google Earth.

13. Operating system concept

An operating system (commonly abbreviated to either OS or O/S) is an interface between hardware and applications; it is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. The operating system acts as a host for applications that are run on the machine. As a host, one of the purposes of an operating

system is to handle the details of the operation of the hardware. This relieves application programs from having to manage these details and makes it easier to write applications. Almost all computers, including handheld computers, desktop computers, supercomputers, and even video game consoles, use an operating system of some type. Some of the oldest models may however use an embedded operating system that may be contained on a compact disk or other data storage device. Operating systems offer a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls. By invoking these interfaces, the application can request a service from the operating system, pass parameters, and receive the results of the operation. Users may also interact with the operating system with some kind of software user interface (UI) like typing commands by using command line interface (CLI) or using a graphical user interface (GUI, commonly pronounced gooey). For hand-held and desktop computers, the user interface is generally considered part of the operating system. On large multi-user systems like UNIX and Unix-like systems, the user interface is generally implemented as an application program that runs outside the operating system. (Whether the user interface should be included as part of the operating system is a point of contention.) Common contemporary operating systems include Microsoft Windows, Mac OS, Linux, BSD and Solaris. Microsoft Windows has a significant majority of market share in the desktop and notebook computer markets, while servers generally run on Linux or other Unix-like systems. Embedded device markets are split amongst several operating systems.

Topic : The User Interface, The File System, And The Iocs

Topic Objective:

At the end of this topic student will be able to understand:

- Understand theAn Operating System's Basic Functions
- Understand the Protected mode and supervisor mode
- Understand the Memory management
- Understand the Virtual memory
- Understand the Multitasking

- Understand the Disk access and file systems
- Understand the Device drivers

Definition/Overview:

An Operating System's Basic Functions: Interrupts are central to operating systems as they provide an efficient way for the operating system to interact and react to its environment.

Protected mode and supervisor mode: Modern CPUs support something called dual mode operation. CPUs with this capability use two modes: protected mode and supervisor mode, which allow certain CPU functions to be controlled and affected only by the operating system kernel.

Memory management: Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs.

Virtual memory: The use of virtual memory addressing (such as paging or segmentation) means that the kernel can choose what memory each program may use at any given time, allowing the operating system to use the same memory locations for multiple tasks.

Multitasking: Multitasking refers to the running of multiple independent computer programs on the same computer; giving the appearance that it is performing the tasks at the same time.

Key Points:

1. An Operating System's Basic Functions.

Interrupts are central to operating systems as they provide an efficient way for the operating system to interact and react to its environment. The alternative is to have the operating system "watch" the various sources of input for events that require action -- not a good use of CPU resources. Interrupt-based programming is directly supported by most CPUs. Interrupts provide a computer with a way of automatically running specific code in response to events. Even very basic computers support hardware interrupts, and allow the programmer to specify code which may be run when that event takes place. When an interrupt is received the computer's hardware automatically suspends whatever program is currently running, saves its status, and runs computer code previously associated with the interrupt. This is analogous to placing a bookmark in a book when someone is interrupted by a phone call and then taking the call. In modern operating systems interrupts are handled by the operating system's kernel.

Interrupts maycome from either the computer's hardware or from the running program. When a hardware device triggers an interrupt the operating system's kernel decides how to deal with this event, generally by running some processing code. How much code gets run depends on the priority of the interrupt (for example: a person usually responds to a smoke detector alarm before answering the phone). The processing of hardware interrupts is a task that is usually delegated to software called device drivers, which may be either part of the operating system's kernel, part of another program, or both. Device drivers may then relay information to a running program by various means. A program may also trigger an interrupt to the operating system. If a program wishes to access hardware for example, it may interrupt the operating system's kernel, which causes control to be passed back to the kernel. The kernel will then process the request. If a program wishes additional resources (or wishes to shed resources) such as memory, it will trigger an interrupt to get the kernel's attention.

2. Protected mode and supervisor mode

Modern CPUs support something called dual mode operation. CPUs with this capability use two modes: protected mode and supervisor mode, which allow certain CPU functions to be controlled and affected only by the operating system kernel. Here, protected mode does not refer specifically to the 80286 (Intel's x86 16-bit microprocessor) CPU feature, although its protected mode is very similar to it. CPUs might have other modes similar to 80286 protected mode as well, such as the virtual 8086 mode of the 80386 (Intel's x86 32-bit microprocessor or i386). However, the term is used here more generally in operating system theory to refer to all modes which limit the capabilities of programs running in that mode, providing things like virtual memory addressing and limiting access to hardware in a manner determined by a program running in supervisor mode. Similar modes have existed in supercomputers, minicomputers, and mainframes as they are essential to fully supporting UNIX-like multi-user operating systems. When a computer first starts up, it is automatically running in supervisor mode. The first few programs to run on the computer, being the BIOS, bootloader and the operating system have unlimited access to hardware - and this is required because, by definition, initializing a protected environment can only be done outside of one. However, when the operating system passes control to another program, it can place the CPU into protected mode. In protected mode, programs may have access to a more limited set of the CPU's instructions. A user program may leave protected mode only by triggering an interrupt, causing control to be passed back to the kernel. In this way the operating system can maintain exclusive control over things like access to hardware and memory. The term "protected mode resource" generally refers to one or more CPU registers, which contain information that the

running program isn't allowed to alter. Attempts to alter these resources generally cause a switch to supervisor mode, where the operating system can deal with the illegal operation the program was attempting (for example, by killing the program).

3. Memory management

Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs. This ensures that a program does not interfere with memory already used by another program. Since programs time share, each program must have independent access to memory. Cooperative memory management, used by many early operating systems assumes that all programs make voluntary use of the kernel's memory manager, and do not exceed their allocated memory. This system of memory management is almost never seen anymore, since programs often contain bugs which can cause them to exceed their allocated memory. If a program fails it may cause memory used by one or more other programs to be affected or overwritten. Malicious programs or viruses may purposefully alter another program's memory or may affect the operation of the operating system itself. With cooperative memory management it takes only one misbehaved program to crash the system. Memory protection enables the kernel to limit a process' access to the computer's memory. Various methods of memory protection exist, including memory segmentation and paging. All methods require some level of hardware support (such as the 80286 MMU) which doesn't exist in all computers. In both segmentation and paging, certain protected mode registers specify to the CPU what memory address it should allow a running program to access. Attempts to access other addresses will trigger an interrupt which will cause the CPU to re-enter supervisor mode, placing the kernel in charge. This is called a segmentation violation or Seg-V for short, and since it is both difficult to assign a meaningful result to such an operation, and because it is usually a sign of a misbehaving program, the kernel will generally resort to terminating the offending program, and will report the error. Windows 3.1-Me had some level of memory protection, but programs could easily circumvent the need to use it. Under Windows 9x all MS-DOS applications ran in supervisor mode, giving them almost unlimited control over the computer. A general protection fault would be produced indicating a segmentation violation had occurred, however the system would often crash anyway.

In most Linux systems, part of the hard disk is reserved for virtual memory when the Operating system is being installed on the system. This part is known as swap space. Windows systems use a swap file instead of a partition.

4. Virtual memory

The use of virtual memory addressing (such as paging or segmentation) means that the kernel can choose what memory each program may use at any given time, allowing the operating system to use the same memory locations for multiple tasks. If a program tries to access memory that isn't in its current range of accessible memory, but nonetheless has been allocated to it, the kernel will be interrupted in the same way as it would if the program were to exceed its allocated memory. (See section on memory management.) Under UNIX this kind of interrupt is referred to as a page fault. When the kernel detects a page fault it will generally adjust the virtual memory range of the program which triggered it, granting it access to the memory requested. This gives the kernel discretionary power over where a particular application's memory is stored, or even whether or not it has actually been allocated yet. In modern operating systems, application memory which is accessed less frequently can be temporarily stored on disk or other media to make that space available for use by other programs. This is called swapping, as an area of memory can be used by multiple programs, and what that memory area contains can be swapped or exchanged on demand.

Further information: Page fault

5. Multitasking

Multitasking refers to the running of multiple independent computer programs on the same computer; giving the appearance that it is performing the tasks at the same time. Since most computers can do at most one or two things at one time, this is generally done via time sharing, which means that each program uses a share of the computer's time to execute. An operating system kernel contains a piece of software called a scheduler which determines how much time each program will spend executing, and in which order execution control should be passed to programs. Control is passed to a process by the kernel, which allows the program access to the CPU and memory. At a later time control is returned to the kernel through some mechanism, so that another program may be allowed to use the CPU. This so-called passing of control between the kernel and applications is called a context switch. An early model which governed the allocation of time to programs was called cooperative multitasking. In this model, when control is passed to a program by the kernel, it may execute for as long as it wants before explicitly returning control to the kernel. This means that a malicious or malfunctioning program may not only prevent any other programs from using

the CPU, but it can hang the entire system if it enters an infinite loop. The philosophy governing preemptive multitasking is that of ensuring that all programs are given regular time on the CPU. This implies that all programs must be limited in how much time they are allowed to spend on the CPU without being interrupted. To accomplish this, modern operating system kernels make use of a timed interrupt. A protected mode timer is set by the kernel which triggers a return to supervisor mode after the specified time has elapsed. (See above sections on Interrupts and Dual Mode Operation.)

On many single user operating systems cooperative multitasking is perfectly adequate, as home computers generally run a small number of well tested programs. Windows NT was the first version of Microsoft Windows which enforced preemptive multitasking, but it didn't reach the home user market until Windows XP, (since Windows NT was targeted at professionals.)

6. Disk access and file systems

Access to files stored on disks is a central feature of all operating systems. Computers store data on disks using files, which are structured in specific ways in order to allow for faster access, higher reliability, and to make better use out of the drive's available space. The specific way in which files are stored on a disk is called a file system, and enables files to have names and attributes. It also allows them to be stored in a hierarchy of directories or folders arranged in a directory tree. Early operating systems generally supported a single type of disk drive and only one kind of file system. Early file systems were limited in their capacity, speed, and in the kinds of file names and directory structures they could use. These limitations often reflected limitations in the operating systems they were designed for, making it very difficult for an operating system to support more than one file system. While many simpler operating systems support a limited range of options for accessing storage systems, operating systems like UNIX and Linux support a technology known as a virtual file system or VFS. An operating system like UNIX supports a wide array of storage devices, regardless of their design or file systems to be accessed through a common application programming interface (API). This makes it unnecessary for programs to have any knowledge about the device they are accessing. A VFS allows the operating system to provide programs with access to an unlimited number of devices with an infinite variety of file systems installed on them through the use of specific device drivers and file system drivers. A connected storage device such as a hard drive is accessed through a device driver. The device driver understands the specific language of the drive and is able to translate that language into a standard language used by the operating system to access all disk drives. On

UNIX this is the language of block devices. When the kernel has an appropriate device driver in place, it can then access the contents of the disk drive in raw format, which may contain one or more file systems. A file system driver is used to translate the commands used to access each specific file system into a standard set of commands that the operating system can use to talk to all file systems. Programs can then deal with these files systems on the basis of filenames, and directories/folders, contained within a hierarchical structure. They can create, delete, open, and close files, as well as gather various information about them, including access permissions, size, and free space, and creation and modification dates. Various differences between file systems make supporting all file systems difficult. Allowed characters in file names, case sensitivity, and the presence of various kinds of file attributes makes the implementation of a single interface for every file system a daunting task. Operating systems tend to recommend the use of (and so support natively) file systems specifically designed for them; for example, NTFS in Windows and ext3 and ReiserFS in Linux. However, in practice, third party drives are usually available to give support for the most widely used filesystems in most general-purpose operating systems (for example, NTFS is available in Linux through NTFS-3g, and ext2/3 and ReiserFS are available in Windows through FS-driver and rfstool).

7. Device drivers

A device driver is a specific type of computer software developed to allow interaction with hardware devices. Typically this constitutes an interface for communicating with the device, through the specific computer bus or communications subsystem that the hardware is connected to, providing commands to and/or receiving data from the device, and on the other end, the requisite interfaces to the operating system and software applications. It is a specialized hardware-dependent computer program which is also operating system specific that enables another program, typically an operating system or applications software package or computer program running under the operating system kernel, to interact transparently with a hardware device, and usually provides the requisite interrupt handling necessary for any necessary asynchronous time-dependent hardware interfacing needs. The key design goal of device drivers is abstraction. Every model of hardware (even within the same class of device) is different. Newer models also are released by manufacturers that provide more reliable or better performance and these newer models are often controlled differently. Computers and their operating systems cannot be expected to know how to control every device, both now and in the future. To solve this problem, OSes essentially dictate how every type of device should be controlled. The function of the device driver is then to translate

these OS mandated function calls into device specific calls. In theory a new device, which is controlled in a new manner, should function correctly if a suitable driver is available. This new driver will ensure that the device appears to operate as usual from the operating systems' point of view for any person.

8. Networking

Currently most operating systems support a variety of networking protocols, hardware, and applications for using them. This means that computers running dissimilar operating systems can participate in a common network for sharing resources such as computing, files, printers, and scanners using either wired or wireless connections. Networks can essentially allow a computer's operating system to access the resources of a remote computer to support the same functions as it could if those resources were connected directly to the local computer. This includes everything from simple communication, to using networked file systems or even sharing another computer's graphics or sound hardware. Some network services allow the resources of a computer to be accessed transparently, such as SSH which allows networked users direct access to a computer's command line interface. Client/server networking involves a program on a computer somewhere which connects via a network to another computer, called a server. Servers, usually running UNIXor Linux, offer (or host) various services to other network computers and users. These services are usually provided through ports or numbered access points beyond the server's network address. Each port number is usually associated with a maximum of one running program, which is responsible for handling requests to that port. A daemon, being a user program, can in turn access the local hardware resources of that computer by passing requests to the operating system kernel. Many operating systems support one or more vendor-specific or open networking protocols as well, for example, SNA on IBM systems, DECnet on systems from Digital Equipment Corporation, and Microsoft-specific protocols (SMB) on Windows. Specific protocols for specific tasks may also be supported such as NFS for file access. Protocols like ESound, or esd can be easily extended over the network to provide sound from local applications, on a remote system's sound hardware.

9. The User Interface.

The **user interface** (also known as Human Computer Interface or Man-Machine Interface (MMI)) is the aggregate of means by which peoplethe usersinteract with the systema

particular machine, device, computer program or other complex tool. The user interface provides means of:

- Input, allowing the users to manipulate a system
- Output, allowing the system to indicate the effects of the users' manipulation.

To work with a system, users have to be able to control the system and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windscreen and exact speed of the vehicle by reading the speedometer. The user interface of the automobile is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile. The term user interface is often used in the context of computer systems and electronic devices. The user interface of a mechanical system, a vehicle or an industrial installation is sometimes referred to as the **Human-Machine Interface** (**HMI**). HMI is a modification of the original term MMI (Man-Machine Interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, but is more commonly used for Human-computer interaction than Human-computer interface. Other terms used are Operator Interface Console (OIC) and Operator Interface Terminal (OIT). However it is abbreviated, the terms refer to the 'layer' that separates a human that is operating a machine from the machine itself. In science fiction, HMI is sometimes used to refer to what is better described as direct neural interface. However, this latter usage is seeing increasing application in the real-life use of (medical) prosthesesthe artificial extension that replaces a missing body part (e.g., cochlear implants). The system may expose several user interfaces to serve different kinds of users. For example, a computerized library database might provide two user interfaces, one for library patrons (limited set of functions, optimized for ease of use) and the other for library personnel (wide set of functions, optimized for efficiency). In some circumstance computers might observe the user, and react according to their actions without specific commands. A means of tracking parts of the body is required, and sensors noting the position of the head, direction of gaze and soon have been used experimentally. This is particularly relevant to immersive interfaces.

10. Usability

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying. Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.

11. User interfaces in computing

In computer science and human-computer interaction, the user interface (of a computer program) refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

Currently (as of 2009) the following types of user interface are the most common:

- Graphical user interfaces (GUI) accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-oriented user interfaces (OOUIs) and application oriented interfaces.
- Web-based user interfaces or web user interfaces (WUI) accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies to provide realtime control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are called Control panel.

User interfaces that are common in various fields outside desktop computing:

- Command line interfaces, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Used for system administration tasks etc.

- Tactile interfaces supplement or replace other forms of output with haptic feedback methods. Used in computerized simulators etc.
- Touch interfaces are graphical user interfaces using a touchscreen display as a combined input and output device. Used in many types of point of sale, industrial processes and machines, self-service machines etc.

Other types of user interfaces:

- Attentive user interfaces manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- Batch interfaces are non-interactive user interfaces, where the user specifies all the details of the batch job in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- Conversational Interface Agents attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- Crossing-based interfaces are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- Gesture interfaces are graphical user interfaces which accept input in a form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- Intelligent user interfaces are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).
- Multi-screen interfaces employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- Noncommand user interfaces, which observe the user to infer his / her needs and intentions, without requiring that he / she formulate explicit commands.
- Object-oriented User Interface (OOUI)
- Reflexive user interfaces where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.

- Tangible user interfaces, which place a greater emphasis on touch and physical environment or its element.
- Text user interfaces are user interfaces which output text, but accept other form of input in addition to or in place of typed command strings.
- Voice user interfaces, which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.
- Natural-Language interfaces - Used for search engines and on webpages. User types in a question and waits for a response.
- Zero-Input interfaces get inputs from a set of sensors instead of querying the user with input dialogs.
- Zooming user interfaces are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail.

12. The File System.

In computing, a **file system** (often also written as **filesystem**) is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. File systems may use a data storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files, they might provide access to data on a file server by acting as clients for a network protocol (e.g., NFS, SMB, or 9P clients), or they may be virtual and exist only as an access method for virtual data (e.g., procfs). It is distinguished from a directory service and registry.

More formally, a file system is a special-purpose database for the storage, organization, manipulation, and retrieval of data.

13. Aspects of file systems

Most file systems make use of an underlying data storage device that offers access to an array of fixed-size blocks, sometimes called sectors, generally a power of 2 in size (512 bytes or 1, 2, or 4 KiB are most common). The file system software is responsible for organizing these sectors into files and directories, and keeping track of which sectors belong to which file and which are not being used. Most file systems address data in fixed-sized units called "clusters" or "blocks" which contain a certain number of disk sectors (usually 1-64). This is the smallest amount of disk space that can be allocated to hold a file. However, file systems need not

make use of a storage device at all. A file system can be used to organize and represent access to any data, whether it be stored or dynamically generated (e.g., procfs).

14. File names

Whether the file system has an underlying storage device or not, file systems typically have directories which associate **file names** with files, usually by connecting the file name to an index in a file allocation table of some sort, such as the FAT in a DOS file system, or an inode in a Unix-like file system. Directory structures may be flat, or allow hierarchies where directories may contain subdirectories. In some file systems, file names are structured, with special syntax for filename extensions and version numbers. In others, file names are simple strings, and per-file metadata is stored elsewhere.

15. Metadata

Other bookkeeping information is typically associated with each file within a file system. The length of the data contained in a file may be stored as the number of blocks allocated for the file or as an exact byte count. The time that the file was last modified may be stored as the file's timestamp. Some file systems also store the file creation time, the time it was last accessed, and the time that the file's meta-data was changed. (Note that many early PC operating systems did not keep track of file times.) Other information can include the file's device type (e.g., block, character, socket, subdirectory, etc.), its owner user-ID and group-ID, and its access permission settings (e.g., whether the file is read-only, executable, etc.). Arbitrary attributes can be associated on advanced file systems, such as XFS, ext2/ext3, some versions of UFS, and HFS+, using extended file attributes. This feature is implemented in the kernels of Linux, FreeBSD and Mac OS X operating systems, and allows metadata to be associated with the file at the file system level. This, for example, could be the author of a document, the character encoding of a plain-text document, or a checksum.

16. Hierarchical file systems

The hierarchical file system was an early research interest of Dennis Ritchie of UNIX fame; previous implementations were restricted to only a few levels, notably the IBM implementations, even of their early databases like IMS. After the success of UNIX, Ritchie extended the file system concept to every object in his later operating system developments, such as Plan 9 and Inferno.

17. Facilities

Traditional file systems offer facilities to create, move and delete both files and directories. They lack facilities to create additional links to a directory (hard links in Unix), rename parent links ("... in Unix-like OS), and create bidirectional links to files. Traditional file

systems also offer facilities to truncate, append to, create, move, delete and in-place modify files. They do not offer facilities to prepend to or truncate from the beginning of a file, let alone arbitrary insertion into or deletion from a file. The operations provided are highly asymmetric and lack the generality to be useful in unexpected contexts. For example, interprocess pipes in UNIX have to be implemented outside of the file system because the pipes concept does not offer truncation from the beginning of files.

18. Secure access

Secure access to basic file system operations can be based on a scheme of access control lists or capabilities. Research has shown access control lists to be difficult to secure properly, which is why research operating systems tend to use capabilities. Commercial file systems still use access control lists.

19. Disk file systems

A disk file system is a file system designed for the storage of files on a data storage device, most commonly a disk drive, which might be directly or indirectly connected to the computer. Examples of disk file systems include FAT (FAT12, FAT16, FAT32, exFAT), NTFS, HFS and HFS+, HPFS, ext2, ext3, ext4, ISO 9660, ODS-5, ZFS and UDF. Some disk file systems are journaling file systems or versioning file systems.

Flash file systems

A flash file system is a file system designed for storing files on flash memory devices. These are becoming more prevalent as the number of mobile devices is increasing, and the capacity of flash memories increase.

While a disk file system can be used on a flash device, this is suboptimal for several reasons:

- Erasing blocks: Flash memory blocks have to be explicitly erased before they can be rewritten. The time taken to erase blocks can be significant, thus it is beneficial to erase unused blocks while the device is idle.
- Random access: Disk file systems are optimized to avoid disk seeks whenever possible, due to the high cost of seeking. Flash memories devices impose no seek latency.
- Wear leveling: Flash memory devices tend to wear out when a single block is repeatedly overwritten; flash file systems are designed to spread out writes evenly.

Log-structured file systems have many of the desirable properties for a flash file system. Such file systems include JFFS2 and YAFFS. A new concept for file management is the concept of a database-based file system. Instead of, or in addition to, hierarchical structured

management, files are identified by their characteristics, like type of file, topic, author, or similar metadata. Each disk operation may involve changes to a number of different files and disk structures. In many cases, these changes are related, meaning that it is important that they all be executed at the same time. Take for example a bank sending another bank some money electronically. The bank's computer will "send" the transfer instruction to the other bank and also update its own records to indicate the transfer has occurred. If for some reason the computer crashes before it has had a chance to update its own records, then on reset, there will be no record of the transfer but the bank will be missing some money. Transaction processing introduces the guarantee that at any point while it is running, a transaction can either be finished completely or reverted completely (though not necessarily both at any given point). This means that if there is a crash or power failure, after recovery, the stored state will be consistent. (Either the money will be transferred or it will not be transferred, but it won't ever go missing "in transit".) This type of file system is designed to be fault tolerant, but may incur additional overhead to do so. Journaling file systems are one technique used to introduce transaction-level consistency to filesystem structures.

20. Network file systems

A network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server. Examples of network file systems include clients for the NFS, AFS, SMB protocols, and file-system-like clients for FTP and WebDAV.

21. Special purpose file systems

A special purpose file system is basically any file system that is not a disk file system or network file system. This includes systems where the files are arranged dynamically by software, intended for such purposes as communication between computer processes or temporary file space. Special purpose file systems are most commonly used by file-centric operating systems such as UNIX. Examples include the procfs (/proc) file system used by some UNIX variants, which grants access to information about processes and other operating system features. Deep space science exploration craft, like Voyager I & II used digital tape-based special file systems. Most modern space exploration craft like Cassini-Huygens used Real-time operating system file systems or RTOS influenced file systems. The Mars Rovers are one such example of an RTOS file system, important in this case because they are implemented in flash memory. Crash counting is a feature of a file system designed as an alternative to journaling. It is claimed that it maintains consistency across crashes without the code complexity of implementing journaling.

22. The Input/Output Control System

Input-output Control is a technique that allows operation to manage facility workflow. It is used to control the size of the queues in front of work centers, thereby helping to control manufacturing lead times. Refer to as "push system" of linking work centers. When a batch of items is completed at one work centre, it is pushed to the next work centre, where it waits in a queue until it is selected to be worked at that work centre. Input-output Control is important because it is a form of queue control, and a great portion of the time that a job spends in a plant is spent waiting in queues. In many job shops and batch manufacturing factories 80 to 95 percent of the total time is queue time. Input-output Control is used to monitor and control the amount if work in a queue at a work centre so that it stays within reasonable bounds. Queue times, therefore, are more consistent and predictable. The actual output, in standard hours of works, that flows from a work centre is the demonstrated actual capacity of the centre. If the capacity differs from the planned amount of work over a few scheduling periods, the problem needs to be investigated. If the work that flows to work centre is greater than the actual output, the queue in front of the centre will grow. On the other words, If the works is arriving faster than it is being processed, we are overloading the facility and a backlog develop. Overloading causes crowding in the facility, leading to inefficiencies and quality problem. If the works is arriving at a slower rate than it is being performed, we are underloading the facility and the work centre may run out of work. Underloading the facility results in idle capacity and wasted resources.

23. The Boot

In computing, **booting** (**booting up**) is a bootstrapping process that starts operating systems when the user turns on a computer system. A **boot sequence** is the initial set of operations that the computer performs when it is switched on. The **bootloader**typically loads the main operating system for the computer.

24. Boot loader

A computer's central processor can only execute program code found in Read-Only Memory (ROM) and Random Access Memory (RAM). Modern operating systems and application program code and data are stored on nonvolatile data storage devices, such as hard disc drives, CD, DVD, USB flash drive, and floppy disk. When a computer is first powered on, it does not have an operating system in ROM or RAM. The computer must initially execute a

small program stored in ROM along with the bare minimum of data needed to access the nonvolatile devices from which the operating system programs and data are loaded into RAM. The small program that starts this sequence of loading into RAM is known as a bootstrap loader, bootstrap or boot loader. This small boot loader program's only job is to load other data and programs which are then executed from RAM. Often, multiple-stage boot loaders are used; during which several programs of increasing complexity sequentially load one after the other in a process of chain loading. This loading may include optional software such as network software, media player, and anti-virus programs. Early computers (such as the PDP-1 through PDP-8 and early models of the PDP-11) had a row of toggle switches on the front panel to allow the operator to manually enter the binary boot instructions into memory before transferring control to the CPU. The boot loader would then read the second-stage boot loader (called Binary Loader of paper tape with checksum) or the operating system in from an outside storage medium such as paper tape, punched card, or a disk drive. Pseudo-assembly code for the bootloader might be as simple as the following eight instructions:

0: set the P register to 8

1: check paper tape reader ready

2: if not ready, jump to 1

3: read a byte from paper tape reader to accumulator

4: if end of tape, jump to 8

5: store accumulator to address in P register

6: increment the P register

7: jump to 1

A related example is based on a loader for a 1970's Nicolet Instrument Corporation minicomputer. Note that the bytes of the second-stage loader are read from paper tape in reverse order.

0: set the P register to 106

1: check paper tape reader ready

2: if not ready, jump to 1

3: read a byte from paper tape reader to accumulator

4: store accumulator to address in P register

5: decrement the P register

6: jump to 1

The length of the second stage loader is such that the final byte overwrites location 6. After the instruction in location 5 executes, location 6 starts the second stage loader executing. The second stage loader then waits for the much longer tape containing the operating system to be placed in the tape reader. The difference between the boot loader and second stage loader is the addition of checking code to trap paper tape read errors, a frequent occurrence with the hardware of the time, which in this case was an ASR-33 teletype. Some computer systems, upon receiving a boot signal from a human operator or a peripheral device, may load a very small number of fixed instructions into memory at a specific location, initialize at least one CPU, and then point the CPU to the instructions and start their execution. These instructions typically start an input operation from some peripheral device (which may be switch-selectable by the operator). Other systems may send hardware commands directly to peripheral devices or I/O controllers that cause an extremely simple input operation (such as "read sector zero of the system device into memory starting at location 1000") to be carried out, effectively loading a small number of bootload instructions into memory; a completion signal from the I/O device may then be used to start execution of the instructions by the CPU. Smaller computers often use less flexible but more automatic bootload mechanisms to ensure that the computer starts quickly and with a predetermined software configuration. In many desktop computers, for example, the bootstrapping process begins with the CPU executing software contained in ROM (for example, the BIOS of an IBM PC) at a predefined address (some CPUs, including the Intel x86 series are designed to execute this software after reset without outside help). This software contains rudimentary functionality to search for devices eligible to participate in booting, and load a small program from a special section (most commonly the boot sector) of the most promising device. It is usually possible to configure the BIOS so that only a certain device can be booted from and/or to give priority to some devices over others (a CD or DVD drive is usually given priority over a hard disk, for instance). Boot loaders may face peculiar constraints, especially in size; for instance, on the IBM PC and compatibles, the first stage of boot loaders located on hard drives must fit into the first 446 bytes of the Master Boot Record, in order to leave room for the 64-byte partition table and the 2-byte 0xAA55 'signature', which the BIOS requires for a proper boot loader. Some operating systems, most notably pre-1995 Macintosh systems from Apple, are so closely interwoven with their hardware that it is impossible to natively boot an operating system other than the standard one. This is the opposite extreme of the bootload using switches mentioned above; it is highly inflexible but relatively error-proof and foolproof as long as all hardware is working normally. A common solution in such situations is to design a

bootloader that works as a program belonging to the standard OS that hijacks the system and loads the alternative OS. This technique was used by Apple for its A/UX UNIX implementation and copied by various freeware operating systems and BeOS Personal Edition 5.

25. Second-stage boot loader

The small program is most often not itself an operating system, but only a second-stage boot loader, such as GRUB, BOOTMGR, LILO or NTLDR. It will then be able to load the operating system properly, and finally transfer execution to it. The system will initialize itself, and may load device drivers and other programs that are needed for the normal operation of the OS.

Many bootloaders (like GRUB, BOOTMGR, LILO, and NTLDR) can be configured to give the user multiple booting choices. These choices can include different operating systems (for dual or multi-booting from different partitions or drives), different kernel versions of the same operating system (e.g., when a newer Linux kernel is installed, this gives one the option of using a known good kernel if problems arise), different kernel options (e.g., booting into a rescue or safe mode) or some standalone program that can function without an operating system, such as memory testers (e.g., memtest86+) or even games. Usually a default choice is preselected with a time delay during which you can press a key to change the choice, after which the default choice is automatically run, so normal booting can occur without interaction. The boot process is considered complete when the computer is ready to interact with the user, or the operating system is capable of running ordinary applications. Typical modern PCs boot in about one minute (of which about 15 seconds are taken by a power-on self test (POST) and a preliminary boot loader, and the rest by loading the operating system); whereas, large servers may take several minutes to boot and start all their services. Many embedded systems must boot immediately. For example, waiting a minute for a digital television or sat-nav to start is generally unacceptable. Therefore such devices have their complete operating system in ROM or flash memory so the device can begin functioning immediately. For these types of embedded system little or no loading is necessary, since the loading can be precomputed and stored on the ROM when the device is made. Large and complex systems may have boot procedures that proceed in multiple phases, each phase loading a more complex version of it, until finally the actual operating system is loaded and ready to execute. Because most operating systems are designed as if they never start or stop, bootload processes sometimes construct a near-snapshot of a running operating system, configure themselves as a mere process within that operating system, and then irrevocably

transfer control into the operating system; the bootload process then terminates normally as any other process would, and the operating system need not have any awareness of the bootload.

26. Flash boot loader

Embedded systems especially in automotive applications rely heavily on Flash Bootloaders to ensure that the ECU (Electronic Control Unit) is programmable either in production or in service. A Flash Bootloader resides in Flash memory, and is always the first application to run after a reset. The Flash bootloader decides whether an application is ready and thereby either stays in the ECU or jumps to the application to start execution. The benefit of having a Flash Bootloader on an ECU is mainly to allow erasing and programming new applications on a single ECU in case of application updates, a recall, or changing a configuration by downloading new calibration files. The most popular Flash Bootloaders are CAN based, i.e. use the Control Area Network protocol to download data to an ECU. These bootloaders use a Diagnostics protocol to communicate and download to an ECU.

In Section 2 of this course you will cover these topics:

- Resource Management
- Ms-Dos Commands
- The Microsoft Windows User Interface
- The Unix/Linux User Interface

Topic : Resource Management

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Memory management
- Understand the Relocation
- Understand the Protection
- Understand the Sharing

- Understand the DOS memory managers
- Understand the Virtual memory

Definition/Overview:

Memory management: Memory management is the act of managing computer memory. In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed. The management of main memory is critical to the computer system.

Relocation: In systems with virtual memory, programs in memory must be able to reside in different parts of the memory at different times.
Protection: Processes should not be able to reference the memory for another process without permission.
Sharing: Even though the memory for different processes is protected from each other, different processes should be able to share information and therefore access the same part of memory.

DOS memory managers: In addition to standard memory management, the 640 KB barrier of MS-DOS and compatible systems led to the development of programs known as memory managers when PC main memories started to be routinely larger than 640 KB in the late 1980s

Key Points:

1. Memory management

Memory management is the act of managing computer memory. In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed. The management of main memory is critical to the computer system. Virtual memory systems separate the memory addresses used by a process from actual physical addresses, allowing separation of processes and increasing the effectively available amount of RAM using disk swapping. The quality of the virtual memory manager can have a big impact on overall system performance. Garbage collection is the automated allocation, and deallocation of computer memory resources for a program. This is generally implemented at the programming language level and is in opposition to manual

memory management, the explicit allocation and deallocation of computer memory resources.

2. Relocation

In systems with virtual memory, programs in memory must be able to reside in different parts of the memory at different times. This is because when the program is swapped back into memory after being swapped out for a while it can not always be placed in the same location. Memory management in the operating system should therefore be able to relocate programs in memory and handle memory references in the code of the program so that they always point to the right location in memory.

3. Protection

Processes should not be able to reference the memory for another process without permission. This is called memory protection, and prevents malicious or malfunctioning code in one program from interfering with the operation of other running programs.

4. Sharing

Even though the memory for different processes is protected from each other, different processes should be able to share information and therefore access the same part of memory.

5. DOS memory managers

In addition to standard memory management, the 640 KB barrier of MS-DOS and compatible systems led to the development of programs known as memory managers when PC main memories started to be routinely larger than 640 KB in the late 1980s (see conventional memory). These move portions of the operating system outside their normal locations in order to increase the amount of conventional or quasi-conventional memory available to other applications. Examples are EMM386, which was part of the standard installation in DOS's later versions, and QEMM. These allowed use of memory above the 640 KB barrier, where memory was normally reserved for RAMs, and high and upper memory.

6. Virtual memory

Virtual memory is a computer system technique which gives an application program the impression that it has contiguous working memory (an address space), while in fact it may be physically fragmented and may even overflow on to disk storage. Systems that use this technique make programming of large applications easier and use real physical memory (e.g. RAM) more efficiently than those without virtual memory. Note that "virtual memory" is more than just "using disk space to extend physical memory size" - that is merely the

extension of the memory hierarchy to include hard disk drives. Extending memory to disk is a normal consequence of using virtual memory techniques, but could be done by other means such as overlays or swapping programs and their data completely out to disk while they are inactive. The definition of "virtual memory" is based on redefining the address space with a contiguous virtual memory addresses to "trick" programs into thinking they are using large blocks of contiguous addresses. All modern general-purpose computer operating systems use virtual memory techniques for ordinary applications, such as word processors, spreadsheets, multimedia players, accounting, etc. Older operating systems, such as DOS and Microsoft Windowsof the 1980s, or those for the mainframes of the 1960s, generally had no virtual memory functionality - notable exceptions being the Atlas, B5000 and Apple Computer's Lisa. Embedded systems and other special-purpose computer systems which require very fast and/or very consistent response times may choose not to use virtual memory due to decreased determinism.

7. Paged virtual memory

Almost all implementations of virtual memory divide the virtual address space of an application program into pages; a page is a block of contiguous virtual memory addresses. Pages are usually at least 4K bytes in size, and systems with large virtual address ranges or large amounts of real memory (e.g. RAM) generally use larger page sizes.

8. Page tables

Almost all implementations use page tables to translate the virtual addresses seen by the application program into physical addresses (also referred to as "real addresses") used by the hardware to process instructions. Each entry in a page table contains the starting virtual address of the page--either the real memory address at which the page is actually stored, or an indicator that the page is currently held in a disk file (if the system uses disk files to let applications use amounts of virtual memory which exceed real memory). Systems can have one page table for the whole system or a separate page table for each application. If there is only one, different applications which are running at the same time share a single virtual address space, i.e. they use different parts of a single range of virtual addresses. Systems which use multiple page tables provide multiple virtual address spaces - concurrent applications think they are using the same range of virtual addresses, but their separate page tables redirect to different real addresses.

9. Paging

Paging is the process of saving inactive virtual memory pages to disk and restoring them to real memory when required. Most virtual memory systems enable programs to use virtual address ranges which in total exceed the amount of real memory (e.g. RAM). To do this they use disk files to save virtual memory pages which are not currently active, and restore them to real memory when they are needed. Pages are not necessarily restored to the same real addresses from which they were saved - applications are aware only of virtual addresses. Usually when a page is going to be restored to real memory, the real memory already contains another virtual memory page which will be saved to disk before the restore takes place.

10. Dynamic address translation

If, while executing an instruction, a CPU fetches an instruction located at a particular virtual address, fetches data from a specific virtual address or stores data to a particular virtual address, the virtual address must be translated to the corresponding physical address. This is done by a hardware component, sometimes called a memory management unit, which looks up the real address (from the page table) corresponding to a virtual address and passes the real address to the parts of the CPU which execute instructions. If the page tables indicate that the virtual memory page is not currently in real memory, the hardware raises a page fault exception (special internal signal) which invokes the paging supervisor component of the operating system (see below).

11. Paging supervisor

This part of the operating system creates and manages the page tables. If the dynamic address translation hardware raises a page fault exception, the paging supervisor searches the page file(s) (on disk) for the page containing the required virtual address, reads it into real physical memory, updates the page tables to reflect the new location of the virtual address and finally tells the dynamic address translation mechanism to start the search again. Usually all of the real physical memory is already in use and the paging supervisor must first save an area of real physical memory to disk and update the page table to say that the associated virtual addresses are no longer in real physical memory but saved on disk. Paging supervisors generally save and overwrite areas of real physical memory which have been least recently used, because these are probably the areas which are used least often. So every time the dynamic address translation hardware matches a virtual address with a real physical memory address, it must put a time-stamp in the page table entry for that virtual address.

12. Permanently resident pages

All virtual memory systems have memory areas that are "pinned down", i.e. cannot be swapped out to secondary storage, for example:

- Interrupt mechanisms generally rely on an array of pointers to the handlers for various types of interrupt (I/O completion, timer event, program error, page fault, etc.). If the pages containing these pointers or the code that they invoke were pageable, interrupt-handling would become even more complex and time-consuming; and it would be especially difficult in the case of page fault interrupts.
- The page tables are usually not pageable.
- Data buffers that are accessed outside of the CPU, for example by peripheral devices that use direct memory access (DMA) or by I/O channels. Usually such devices and the buses (connection paths) to which they are attached use physical memory addresses rather than virtual memory addresses. Even on buses with an IOMMU, which is a special memory management unit that can translate virtual addresses used on an I/O bus to physical addresses, the transfer cannot be stopped if a page fault occurs and then restarted when the page fault has been processed. So pages containing locations to which or from which a peripheral device is transferring data are either permanently pinned down or pinned down while the transfer is in progress.
- Timing-dependent kernel/application areas cannot tolerate the varying response time caused by paging.

13. Virtual=real operation

In MVS, z/OS, and similar OSes, some parts of the systems memory are managed in virtual=real mode, where every virtual address corresponds to a real address. Those are:

- interrupt mechanisms
- paging supervisor and page tables
- all data buffers accessed by I/O channels
- application programs which use non-standard methods of managing I/O and therefore provide their own buffers and communicate directly with peripherals (programs that create their own channel command words).

In IBM's early virtual memory operating systems virtual=real mode was the only way to "pin down" pages. z/OS has 3 modes, V=V (virtual=virtual; fully pageable), V=R and V=F (virtual = fixed, i.e. "pinned down" but with DAT operating).

14. Segmented virtual memory

Some systems, such as the Burroughs large systems, do not use paging to implement virtual memory. Instead, they use segmentation, so that an application's virtual address space is divided into variable-length segments. A virtual address consists of a segment number and an offset within the segment. Memory is still physically addressed with a single number (called absolute or linear address). To obtain it, the processor looks up the segment number in a segment table to find a segment descriptor. The segment descriptor contains a flag indicating whether the segment is present in main memory and, if it is, the address in main memory of the beginning of the segment (segment's base address) and the length of the segment. It checks whether the offset within the segment is less than the length of the segment and, if it isn't, an interrupt is generated. If a segment is not present in main memory, a hardware interrupt is raised to the operating system, which may try to read the segment into main memory, or to swap in. The operating system might have to remove other segments (swap out) from main memory in order to make room in main memory for the segment to be read in. Notably, the Intel 80286 supported a similar segmentation scheme as an option, but it was unused by most operating systems. It is possible to combine segmentation and paging, usually dividing each segment into pages. In systems that combine them, such as Multics and the IBM System/38 and IBM System i machines, virtual memory is usually implemented with paging, with segmentation used to provide memory protection. With the Intel 80386 and later IA-32 processors, the segments reside in a 32-bit linear paged address space, so segments can be moved into and out of that linear address space, and pages in that linear address space can be moved in and out of main memory, providing two levels of virtual memory; however, few if any operating systems do so. Instead, they only use paging. The difference between virtual memory implementations using pages and using segments is not only about the memory division with fixed and variable sizes, respectively. In some systems, e.g. Multics, or later System/38 and Prime machines, the segmentation was actually visible to the user processes, as part of the semantics of a memory model. In other words, instead of a process just having a memory which looked like a single large vector of bytes or words, it was more structured. This is different from using pages, which doesn't change the model visible to the process. This had important consequences. A segment wasn't just a "page with a variable length", or a simple way to lengthen the address space (as in Intel 80286). In

Multics, the segmentation was a very powerful mechanism that was used to provide a single-level virtual memory model, in which there was no differentiation between "process memory" and "file system" - a process' active address space consisted only a list of segments (files) which were mapped into its potential address space, both code and data. It is not the same as the later mmap function in Unix, because inter-file pointers don't work when mapping files into semi-arbitrary places. Multics had such addressing mode built into most instructions. In other words it could perform relocated inter-segment references, thus eliminating the need for a linker completely. This also worked when different processes mapped the same file into different places in their private address spaces.

15. Multi-tasking

In computing, multitasking is a method by which multiple tasks, also known as processes, share common processing resources such as a CPU. In the case of a computer with a single CPU, only one task is said to be running at any point in time, meaning that the CPU is actively executing instructions for that task. Multitasking solves the problem by scheduling which task may be the one running at any given time, and when another waiting task gets a turn. The act of reassigning a CPU from one task to another one is called a context switch. When context switches occur frequently enough the illusion of parallelism is achieved. Even on computers with more than one CPU (called multiprocessor machines), multitasking allows many more tasks to be run than there are CPUs.
Operating systems may adopt one of many different scheduling strategies, which generally fall into the following categories:

- In multiprogramming systems, the running task keeps running until it performs an operation that requires waiting for an external event (e.g. reading from a tape) or until the computer's scheduler forcibly swaps the running task out of the CPU. Multiprogramming systems are designed to maximize CPU usage.
- In time-sharing systems, the running task is required to relinquish the CPU, either voluntarily or by an external event such as a hardware interrupt. Time sharing systems are designed to allow several programs to execute apparently simultaneously.
- In real-time systems, some waiting tasks are guaranteed to be given the CPU when an external event occurs. Real time systems are designed to control mechanical devices such as industrial robots, which require timely processing.

- The term time-sharing is no longer commonly used, having been replaced by simply multitasking.

16. Multiprogramming

In the early days of computing, CPU time was expensive, and peripherals were very slow. When the computer ran a program that needed access to a peripheral, the CPU would have to stop executing program instructions while the peripheral processed the data. This was deemed very inefficient. The first efforts to create multiprogramming systems took place in the 1960s. Several different programs in batch were loaded in the computer memory, and the first one began to run. When the first program reached an instruction waiting for a peripheral, the context of this program was stored away, and the second program in memory was given a chance to run. The process continued until all programs finished running. Multiprogramming doesn't give any guarantee that a program will run in a timely manner. Indeed, the very first program may very well run for hours without needing access to a peripheral. As there were no users waiting at an interactive terminal, this was no problem: users handed a deck of punched cards to an operator, and came back a few hours later for printed results. Multiprogramming greatly reduced the waiting. The early OS/360 primary control program (PCP) followed the above model but was replaced the very next year, 1967, by MFT which limited the amount of CPU time any single process could consume before being switched out.

17. Cooperative multitasking/time-sharing

When computer usage evolved from batch mode to interactive mode, multiprogramming was no longer a suitable approach. Each user wanted to see his program running as if it was the only program in the computer. The use of time sharing made this possible, with the qualification that the computer would not seem as fast to any one user as it really would be if it were running only that user's program. Early multitasking systems consisted of suites of related applications that voluntarily ceded time to each other. This approach, which was eventually supported by many computer operating systems, is today known as cooperative multitasking. Although it is rarely used in larger systems, Microsoft Windows prior to Windows 95 and Windows NT, and Mac OS prior to Mac OS X both used cooperative multitasking to enable the running of multiple applications simultaneously. Windows 9x also used cooperative multitasking, but only for 16-bit legacy applications, much the same way as pre-Leopard PowerPC versions of Mac OS X used it for Classic applications. Cooperative

multitasking is still used today on RISC OS systems. Because a cooperatively multitasked system relies on each process to regularly give time to other processes on the system, one poorly designed program can cause the whole system to hang.

18. Preemptive multitasking/time-sharing

Preemptive multitasking allows the computer system to more reliably guarantee each process a regular "slice" of operating time. It also allows the system to rapidly deal with important external events like incoming data, which might require the immediate attention of one or another process. Operating systems were developed to take advantage of these hardware capabilities and run multiple processes preemptively. For example, preemptive multitasking was implemented in the earliest version of UNIX in 1969, and is standard in UNIX and Unix-like operating systems, including Linux, Solaris and BSD with its derivatives. At any specific time, processes can be grouped into two categories: those that are waiting for input or output (called "I/O bound"), and those that are fully utilizing the CPU ("CPU bound"). In primitive systems, the software would often "poll", or "busywait" while waiting for requested input (such as disk, keyboard or network input). During this time, the system was not performing useful work. With the advent of interrupts and preemptive multitasking, I/O bound processes could be "blocked", or put on hold, pending the arrival of the necessary data, allowing other processes to utilize the CPU. As the arrival of the requested data would generate an interrupt, blocked processes could be guaranteed a timely return to execution. The earliest preemptive multitasking OS available to home users was Sinclair QDOS on the Sinclair QL released in 1984. Preemptive multitasking was later adopted on theApple Macintosh by Mac OS 9.x as an additional API, i.e. the application could be programmed to use the preemptive or co-operative model, and all legacy applications were multitasked cooperatively within a single process. Mac OS X, being a Unix-like system, uses preemptive multitasking for all native applications, although Classic applications may be multitasked cooperatively as they run in fact under Mac OS 9 running as OS X process. A similar model is used in Windows 9x and Windows NT family, where native 32-bit applications are multitasked preemptively, and legacy 16-bit Windows 3.x are multitasked cooperatively within a single process, although in the NT family it is possible to force 16-bit application to run as a separate preemptively multitasked process. 64-bit editions of Windows, both for the x86-64 and Itanium architectures no longer provide support for legacy 16-bit applications, and thus provide preemptive multitasking for all supported applications.

19. Real time

Another reason for multitasking was in the design of real-time computing systems, where there are a number of possibly unrelated external activities needed to be controlled by a single processor system. In such systems a hierarchical interrupt system was coupled with process prioritization to ensure that key activities were given a greater share of available process time.

20. Multithreading

As multitasking greatly improved the throughput of computers, programmers started to implement applications as sets of cooperating processes (e.g. one process gathering input data, one process processing input data, one process writing out results on disk.) This, however, required some tools to allow processes to efficiently exchange data. Threads were born from the idea that the most efficient way for cooperating processes to exchange data would be to share their entire memory space. Thus, threads are basically processes that run in the same memory context. Threads are described as lightweight because switching between threads does not involve changing the memory context. While threads are scheduled preemptively, some operating systems provide a variant to threads, named fibers that are scheduled cooperatively. On operating systems that do not provide fibers, an application may implement its own fibers using repeated calls to worker functions. Fibers are even more lightweight than threads, and somewhat easier to program with, although they tend to lose some or all of the benefits of threads on machines with multiple processors.

Some systems directly support multithreading in hardware.

21. Memory protection

When multiple programs are present in memory, an ill-behaved program may (inadvertently or deliberately) overwrite memory belonging to another program, or even to the operating system itself. The operating system therefore restricts the memory accessible to the running program. A program trying to access memory outside its allowed range is immediately stopped before it can change memory belonging to another process. Another key innovation was the idea of privilege levels. Low privilege tasks are not allowed some kinds of memory access and are not allowed to perform certain instructions. When a task tries to perform a privileged operation a trap occurs and a supervisory program running at a higher level is allowed to decide how to respond. This created the possibility of virtualizing the entire system, including virtual peripheral devices. Such a simulation is called a virtual machine operating system. Early virtual machine systems did not have virtual memory, but both are common today.

22. Memory swapping

Use of a swap file or swap partition is a way for the operating system to provide more memory than is physically available by keeping portions of the primary memory in secondary storage. While multitasking and memory swapping are two completely unrelated techniques, they are very often used together, as swapping memory allows more tasks to be loaded at the same time. Typically, a multitasking system allows another process to run when the running process hits a point where it has to wait for some portion of memory to be reloaded from secondary storage.

23. Programming in a multitasking environment

Processes that are entirely independent are not much trouble to program. Most of the complexity in multitasking systems comes from the need to share computer resources between tasks and to synchronize the operation of co-operating tasks. Various concurrent computing techniques are used to avoid potential problems caused by multiple tasks attempting to access the same resource. Bigger computer systems were sometimes built with a central processor(s) and some number of I/O processors, a kind of asymmetric multi-processing. Over the years, multitasking systems have been refined. Modern operating systems generally include detailed mechanisms for prioritizing processes, while symmetric multiprocessing has introduced new complexities and capabilities.

24. Time-sharing

Time-sharingrefers to sharing a computing resource among many users by multitasking. Its introduction in the 1960s, and emergence as the prominent model of computing in the 1970s, represents a major historical shift in the history of computing. By allowing a large number of users to interact simultaneously on a single computer, time-sharing dramatically lowered the cost of providing computing, while at the same time making the computing experience much more interactive.

25. Match processing

The earliest computers were extremely expensive devices, and very slow. Machines were typically dedicated to a particular set of tasks and operated by control panel, the operator manually entering small programs via switches in order to load and run other programs. These programs might take hours, even weeks, to run. As computers grew in speed, run times

dropped, and suddenly the time taken to start up the next program became a concern. The batch processing methodologies evolved to decrease these dead times, cuing up programs so as soon as one completed the next would start. To support a batch processing operation, a number of card punch or paper tape writers would be used by programmers, who would use these inexpensive machines to write their programs "offline". When they completed typing them, they were submitted to the operations team, who would schedule them for running. Important programs would be run quickly, less important ones were unpredictable. When the program was finally run, the output, generally printed, would be returned to the programmer. The complete process might take days, during which the programmer might never see the computer. The alternative, allowing the user to operate the computer directly, was generally far too expensive to consider. This was because the user had long delays where they were simply sitting there entering code. This limited developments in direct interactivity to organizations that could afford to waste computing cycles, large universities for the most part. Programmers at the universities decried the inhumanist behaviors that batch processing imposed, to the point that Stanford students made a short film humorously critiquing it. They experimented with new ways to directly interact with the computer, a field today known as human-computer interaction.

26. Time-sharing

Time-sharing developed out of the realization that while any single user was inefficient, a large group of users together were not. This was due to the pattern of interaction; in most cases users entered bursts of information followed by long pause, but a group of users working at the same time would mean that the pauses of one user would be used up by the activity of the others. Given an optimal group size, the overall process could be very efficient. Similarly, small slices of time spent waiting for disk, tape, or network input could be granted to other users. Implementing a system able to take advantage of this would be difficult. Batch processing was really a methodological development on top of the earliest systems; computers still ran single programs for single users at any time, all that batch processing changed was the time delay between one program and the next. Developing a system that supported multiple users at the same time was a completely different concept, the "state" of each user and their programs would have to be kept in the machine, and then switch between them quickly. This would take up computer cycles, and on the slow machines of the era this was a concern. However, as computers rapidly improved in speed, and especially size of core memory to keep the state, the overhead of time-sharing continually reduced in overall terms.

The concept was first described publicly in early 1957 by Bob Bemer as part of an article in Automatic Control Magazine. The first project to implement a time-sharing system was initiated by John McCarthy in late 1957, on a modified IBM 704, and later an additionally modified IBM 7090 computer. Although he left to work on Project MAC and other projects, one of the results of the project, known as the Compatible Time-Sharing System or CTSS, was demonstrated in November 1961. CTSS has a good claim to be the first time-sharing system and remained in use until 1973. The first commercially successful time-sharing system was the Dartmouth Time-Sharing System (DTSS) which was first implemented at Dartmouth Collegein 1964 and subsequently formed the basis of General Electric's computer bureau services. DTSS influenced the design of other early timesharing systems developed by Hewlett Packard, Control Data Corporation, UNIVAC and others (in addition to introducing the BASIC programming language).

27. Evolution

Throughout the late 1960s and the 1970s, computer terminals were multiplexed onto large institutional mainframe computers (central computer systems), which in many implementations sequentially polled the terminals to see if there was any additional data or action requested by the computer user. Later technology in interconnections was interrupt driven, and some of these used parallel data transfer technologies like, for example, the IEEE 488 standard. Generally, computer terminals were utilized on college properties in much the same places as desktop computers or personal computers are found today. In the earliest days of personal computers, many were in fact used as particularly smart terminals for time-sharing systems. With the rise of microcomputing in the early 1980s, time-sharing faded into the background because the individual microprocessors were sufficiently inexpensive that a single person could have all the CPU time dedicated solely to their needs, even when idle. The Internet has brought the general concept of time-sharing back into popularity. Expensive corporate server farms costing millions can host thousands of customers all sharing the same common resources. As with the early serial terminals, websites operate primarily in bursts of activity followed by periods of idle time. This bursting nature permits the service to be used by many website customers at once, and none of them notice any delays in communications until the servers start to get very busy.

28. Time-sharing business

In the 1960s, several companies started providing time-sharing services as service bureaus. Early systems used Teletype K/ASR-33s or K/ASR-35s in ASCII environments, and IBM

Selectric typewriter-based terminals in EBCDIC environments. They would connect to the central computer by dial-up Bell 103A modem or acoustically coupled modems operating at 10-15 characters per second. Later terminals and modems supported 30-120 characters per second. The time-sharing system would provide a complete operating environment, including a variety of programming language processors, various software packages, file storage, bulk printing, and off-line storage. Users were charged rent for the terminal, a charge for hours of connect time, a charge for seconds of CPU time, and a charge for kilobyte-months of disk storage. Common systems used for time-sharing included the SDS 940, the PDP-10, and the IBM 360. Companies providing this service included GE's GEISCO, IBM subsidiary The Service Bureau Corporation, Tymshare (founded in 1966), National CSS (founded in 1967 and bought by Dun & Bradstreet in 1979), Dial Data (bought by Tymshare in 1968), and Bolt, Beranek, and Newman. By 1968, there were 32 such service bureaus serving the NIH alone. The Auerbach Guide to Timesharing 1973 edition lists 125 different timesharing services using equipment from Burroughs, CDC, DEC, HP, Honeywell, IBM, RCA, Univac and XDS.

29. The Computer Utility

A great deal of thought was given in the 1970s to centralized computer resources being offered as computing utilities, the same as the electrical or telephone utilities. Ted Nelson's original "Xanadu" hypertext repository was envisioned as such a service. It became clear as the computer industry grew that no such consolidation of computing resources would occur as timesharing systems. Some argue that the move through client-server computing to centralized server farms and virtualization presents a market for computing utilities again.

30. The Virtual Machine Concept

In computer science, a virtual machine (VM) is a software implementation of a machine (computer) that executes programs like a real machine. A virtual machine was originally defined by Popek and Goldberg as "an efficient, isolated duplicate of a real machine". Current use includes virtual machines which have no direct correspondence to any real hardware. Virtual machines are separated into two major categories, based on their use and degree of correspondence to any real machine. A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS). In contrast, a process virtual machine is designed to run a single program, which means that it supports a single process. An essential characteristic of a virtual machine is that the software

running inside is limited to the resources and abstractions provided by the virtual machine -- it cannot break out of its virtual world. Example: A program written in Java receives services from the Java Runtime Environment (JRE) software by issuing commands to, and receiving the expected results from, the Java software. By providing these services to the program, the Java software is acting as a "virtual machine", taking the place of the operating system or hardware for which the program would ordinarily be tailored.

31. System virtual machines

System virtual machines (sometimes called hardware virtual machines) allow the sharing of the underlying physical machine resources between different virtual machines, each running its own operating system. The software layer providing the virtualization is called a virtual machine monitor or hypervisor. A hypervisor can run on bare hardware (Type 1 or native VM) or on top of an operating system (Type 2 or hosted VM).

The main advantages of system VMs are:

- multiple OS environments can co-exist on the same computer, in strong isolation from each other
- the virtual machine can provide an instruction set architecture (ISA) that is somewhat different from that of the real machine

Multiple VMs each running their own operating system (called guest operating system) are frequently used in server consolidation, where different services that used to run on individual machines in order to avoid interference are instead run in separate VMs on the same physical machine. This use is frequently called quality-of-service isolation (QoS isolation).

The desire to run multiple operating systems was the original motivation for virtual machines, as it allowed time-sharing a single computer between several single-tasking OSes.

The guest OSes do not have to be all the same, making it possible to run different OSes on the same computer (e.g., Microsoft Windows and Linux, or older versions of an OS in order to support software that has not yet been ported to the latest version). The use of virtual machines to support different guest OSes is becoming popular in embedded systems; a typical use is to support a real-time operating system at the same time as a high-level OS such as Linux or Windows. Another use is to sandbox an OS that is not trusted, possibly because it is a system under development. Virtual machines have other advantages for OS development, including better debugging access and faster reboots. Alternate techniques such as Solaris

Zones provide a level of isolation within a single operating system. This does not have isolation as complete as a VM. A kernel exploit in a system with multiple zones will affect all zones. Achieving the same goal in a virtual machine implementation would require exploiting a weakness in the hypervisor. A hypervisor typically has a smaller "attack surface" than a complete operating system, making this more challenging. Further, a kernel exploit in a VM guest would not affect other VMs on the host, just as a successful intrusion into one zone would not necessarily affect other zones. Zones are not virtual machines, but an example of "operating-system virtualization". This includes other "virtual environments" (also called "virtual servers") such as Virtuozzo, FreeBSD Jails, Linux-VServer, chroot jail, and OpenVZ. These provide some form of encapsulation of processes within an operating system. These technologies have the advantages of being more resource-efficient than full virtualization and having better observability into multiple guests simultaneously; the disadvantage is that, generally, they can only run a single operating system and a single version/patch level of that operating system - so, for example, they cannot be used to run two applications, one of which only supports a newer OS version and the other only supporting an older OS version on the same hardware. However, Sun Microsystems has enhanced Solaris Zones to allow some zones to behave like Solaris 8 or Solaris 9 systems by adding a system call translator.

31. Process virtual machines

A process VM, sometimes called an application virtual machine, runs as a normal application inside an OS and supports a single process. It is created when that process is started and destroyed when it exits. Its purpose is to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system, and allows a program to execute in the same way on any platform. A process VM provides a high-level abstraction that of a high-level programming language (compared to the low-level ISA abstraction of the system VM). Process VMs are implemented using an interpreter; performance comparable to compiled programming languages is achieved by the use of just-in-time compilation. This type of VM has become popular with the Java programming language, which is implemented using the Java virtual machine. Another example is the .NET Framework, which runs on a VM called the Common Language Runtime. A special case of process VMs are systems that abstract over the communication mechanisms of a (potentially heterogeneous) computer cluster. Such a VM does not consist of a single process, but one process per physical machine in the cluster. They are designed to ease the task of programming parallel applications by letting the programmer focus on algorithms rather than the communication mechanisms provided by the interconnect and the OS. They do not hide

the fact that communication takes place, and as such do not attempt to present the cluster as a single parallel machine. Unlike other process VMs, these systems do not provide a specific programming language, but are embedded in an existing language; typically such a system provides bindings for several languages (e.g., C and FORTRAN). Examples are PVM (Parallel Virtual Machine) and MPI (Message Passing Interface). They are not strictly virtual machines, as the applications running on top still have access to all OS services, and are therefore not confined to the system model provided by the "VM".

32. Techniques

This approach is described as full virtualization of the hardware, and can be implemented using a Type 1 or Type 2 hypervisor. (A Type 1 hypervisor runs directly on the hardware; a Type 2 hypervisor runs on another operating system, such as Linux). Each virtual machine can run any operating system supported by the underlying hardware. Users can thus run two or more different "guest" operating systems simultaneously, in separate "private" virtual computers. The pioneer system using this concept was IBM's CP-40, the first (1967) version of IBM's CP/CMS (1967-1972) and the precursor to IBM's VM family (1972-present). With the VM architecture, most users run a relatively simple interactive computing single-user operating system, CMS, as a "guest" on top of the VM control program (VM-CP). This approach kept the CMS design simple, as if it were running alone; the control program quietly provides multitasking and resource management services "behind the scenes". In addition to CMS, VM users can run any of the other IBM operating systems, such as MVS or z/OS. z/VM is the current version of VM, and is used to support hundreds or thousands of virtual machines on a given mainframe. Some installations use Linux for zSeries to run Web servers, where Linux runs as the operating system within many virtual machines. Full virtualization is particularly helpful in operating system development, when experimental new code can be run at the same time as older, more stable, versions, each in a separate virtual machine. The process can even be recursive: IBM debugged new versions of its virtual machine operating system, VM, in a virtual machine running under an older version of VM, and even used this technique to simulate new hardware. The standard x86 processor architecture as used in modern PCs does not actually meet the Popek and Goldberg virtualization requirements. Notably, there is no execution mode where all sensitive machine instructions always trap, which would allow per-instruction virtualization. Despite these limitations, several software packages have managed to provide virtualization on the x86 architecture, even though dynamic recompilation of privileged code, as first implemented by VMware, incurs some performance overhead as compared to a VM running on a natively

virtualizable architecture such as the IBM System/370 or Motorola MC68020. By now, several other software packages such as Virtual PC, VirtualBox, Parallels Workstation and Virtual Iron manage to implement virtualization on x86 hardware.

Intel and AMD have introduced features to their x86 processors to enable virtualization in hardware.

33. Emulation of a non-native system

Virtual machines can also perform the role of an emulator, allowing software applications and operating systems written for another computer processor architecture to be run.

Some virtual machines emulate hardware that only exists as a detailed specification. For example:

- One of the first was the p-code machine specification, which allowed programmers to write Pascal programs that would run on any computer running virtual machine software that correctly implemented the specification.
- The specification of the Java virtual machine.
- The Common Language Infrastructure virtual machine at the heart of the Microsoft .NET initiative.
- Open Firmware allows plug-in hardware to include boot-time diagnostics, configuration code, and device drivers that will run on any kind of CPU.

This technique allows diverse computers to run any software written to that specification; only the virtual machine software itself must be written separately for each type of computer on which it runs.

34. Operating system-level virtualization

Operating System-level Virtualization is a server virtualization technology which virtualizes servers on an operating system (kernel) layer. It can be thought of as partitioning: a single physical server is sliced into multiple small partitions (otherwise called virtual environments (VE), virtual private servers (VPS), guests, zones, etc.); each such partition looks and feels like a real server, from the point of view of its users. For example, Solaris Zones supports multiple guest OSes running under the same OS (such as Solaris 10). All guest OSes have to use the same kernel level and cannot run as different OS versions. Solaris native Zones also requires that the host OS be a version of Solaris; other OSes from other manufacturers are not supported., however you need to use Solaris Branded zones to use another OSes as zones. Another example is AIX, which provides the same technique under the name of Micro

Partitioning. The operating system level architecture has low overhead that helps to maximize efficient use of server resources. The virtualization introduces only a negligible overhead and allows running hundreds of virtual private servers on a single physical server. In contrast, approaches such as pure virtualization (like VMware) and paravirtualization (like Xen or UML) cannot achieve such level of density, due to overhead of running multiple kernels. From the other side, operating system-level virtualization does not allow running different operating systems (i.e. different kernels), although different libraries, distributions etc. are possible.

35. Peripheral Device Management

A peripheral is a device attached to a host computer behind the chipset whose primary functionality is dependent upon the host, and can therefore be considered as expanding the hosts capabilities, while not forming part of the system's core architecture. Some of the more common peripheral devices are printers, scanners, disk drives, tape drives, microphones, speakers, and cameras. Peripheral devices can also include other computers on a network system. A device can also refer to a non-physical item, such as a pseudo terminal, a RAM drive, or a virtual network adapter. Some people do not consider internal devices such as video capture cards to be peripherals because they are added inside the computer case; for them, the term peripherals are reserved exclusively for devices that are hooked up externally to the computer. It is debatable however whether PCMCIA cards qualify as peripherals under this restrictive definition, because some of them go fully inside the laptop, while some, like Wi-Fi cards, have external appendages. The term is different from computer accessories: Computer peripheral has a narrow meaning that refers only to the input output devices of a computer, whereas, computer accessories has a broader meaning, that refers, all the parts that support a computer which includes motherboards, sensors, chips, including all the input and output devices.

Topic : Ms-Dos Commands

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the MS-DOS
- Understand the Getting Started
- Understand the File System
- Understand the Aspects of file systems
- Understand the File names
- Understand the Metadata
- Understand the Hierarchical file systems
- Understand the Facilities
- Understand the Pipeline
- Understand the Multiprocessed pipelines

Definition/Overview:

MS-DOS: MS-DOS (short for Microsoft **D**isk **O**perating **S**ystem) is an operating system commercialized by Microsoft. It was the most commonly used member of the DOS family of operating systems and was the main operating system for personal computers during the 1980s.

The File System: In computing, a file system (often also written as filesystem) is a method for storing and organizing computer files and the data they contain to make it easy to find and access them.

Aspects of file systems: Most file systems make use of an underlying data storage device that offers access to an array of fixed-size blocks, sometimes called sectors, generally a power of 2 in size (512 bytes or 1, 2, or 4 KiB are most common).

File names: Whether the file system has an underlying storage device or not, file systems typically have directories which associate file names with files, usually by connecting the file

name to an index in a file allocation table of some sort, such as the FAT in a DOS file system, or an inode in a Unix-like file system.

Metadata: Other bookkeeping information is typically associated with each file within a file system.

Key Points:

1. MS-DOS

MS-DOS (short for **M**icro**s**oft**D**isk **O**perating **S**ystem) is an operating system commercialized by Microsoft. It was the most commonly used member of the DOS family of operating systems and was the main operating system for personal computers during the 1980s. It was based on the Intel 8086 family of microprocessors, particularly the IBM PC and compatibles. It was gradually replaced on consumer desktop computers by operating systems offering a graphical user interface (GUI), in particular by various generations of the Microsoft Windows operating system and Linux. MS-DOS was known before as QDOS (Quick and Dirty Operating System) and 86-DOS. MS-DOS development originally started in 1981,and was first released in 1982 as MS-DOS 1.0. Several versions were released under different names for different hardware. MS-DOS had eight major versions released before Microsoft stopped development in 2000. It was the key product in Microsoft's growth from a programming languages company to a diverse software development firm, providing the company with essential revenue and marketing resources. It was also the underlying basic operating system on which early versions of Windows ran as a GUI. MS-DOS was a renamed form of 86-DOS (informally known as the Quick-and-Dirty Operating System or Q-DOS) owned by Seattle Computer Products, written by Tim Paterson. Microsoft needed an operating system for the then-new Intel 8086 but it had none available, so it licensed 86-DOS and released a version of it as MS-DOS 1.0.Development started on 1981, and MS-DOS 1.0 was released with the IBM PC on 1982. Tim Paterson is considered the original author of DOS and he is called "The Father of DOS". Worried by possible legal problems, in June 1981 Microsoft made an offer to Rod Brock, the owner of Seattle Computer, to buy the rights for 86-DOS. An agreement to release all rights to the software was signed on June 1981. The total cost was $75,000. Originally MS-DOS was designed to be an operating system that could run on any 8086-family computer. Each computer would have its own distinct hardware and its own version of MS-DOS. The greater speed attainable by direct control of hardware was of particular importance when running computer games. IBM-compatible architecture then

became the goal. Soon all 8086-family computers closely emulated IBM's hardware, and a single version of MS-DOS was all that was needed for the market. While MS-DOS appeared on PC clones, true IBM computers used PC DOS, a rebranded form of MS-DOS. Incidentally, the dependence on IBM-compatible hardware caused major problems for the computer industry when the original design had to be changed. For example, the original design could support no more than 640 kilobytes of memory. Manufacturers had to develop complicated schemes to access additional memory. This would not have been a limitation if the original idea of interfacing with hardware through MS-DOS had endured.

2. Getting Started

On microcomputers based on the Intel 8086 and 8088 processorsincluding but by no means restricted to the IBM PC (and clones) architecturethe initial competition to the PC DOS/MS-DOS line came from Digital Research, whose CP/M operating system had inspired MS-DOS. Digital Research released CP/M-86 a few months after MS-DOS, and it was offered as an alternative to MS-DOS and Microsoft's licensing requirements, but at a higher price. Executable programs for CP/M-86 and MS-DOS were not interchangeable with each other; much applications software was sold in both MS-DOS and CP/M-86 versions until MS-DOS became preponderant (later Digital Research operating systems could run both MS-DOS and CP/M-86 software). MS-DOS supported the simple .COM and the more advanced relocatable .EXE executable file formats; CP/M-86 a relocatable format using the file extension .CMD. In the later days of MS-DOS, once the IBM-compatible platform was chosen, one would buy any PC clone, get any copy of MS-DOS (or IBM-branded MS-DOS, i.e., PC DOS)done. Many people do not realize that in the early days one chose an IBM PC, a Sirius, a Kaypro, or an Apricot, or other make; these machines all had different architecture and did not even accept the same expansion cards; many of them were not limited to a maximum of 640 kilobytes of system memory, unlike the PC and clones. Then the decision whether to use MS-DOS or CP/M-86 had to be taken, and the appropriate one had to be acquired from the computer manufacturer; it was not possible to use versions of MS-DOS, or PC DOS, interchangeably. One was then tied to using software for the operating system chosen (or, funds permitting, having floppy disks for both, and booting the appropriate one). In the business world the 808x-based machines that MS-DOS was tied to face competition from the UNIX operating system which ran on many different hardware architectures. Microsoft itself sold a version of UNIXfor the PC called Xenix.

In the emerging world of home users, a variety of other computers based on various other processors were in serious competition with the IBM PC: the Apple II, early Apple Macintosh, the Commodore 64 and others did not use the 808x processor; many 808x machines of different architectures used custom versions of MS-DOS. At first all these machines were in competition. In time the IBM PC hardware configuration became dominant in the 808x market as software written to communicate directly with the PC hardware without using standard operating system calls ran much faster, but on true PC-compatibles only. Non-PC-compatible 808x machines were too small a market to have fast software written for them alone, and the market remained open only for IBM PCs and machines that closely imitated their architecture, all running either a single version of MS-DOS compatible only with PCs, or the equivalent IBM PC DOS. Most clones cost much less than IBM-branded machines of similar performance, and became widely used by home users, while IBM PCs had a large share of the business computer market. Microsoft and IBM together began what was intended as the follow-on to MS/PC DOS, called OS/2. When OS/2 was released in 1987, Microsoft began an advertising campaign announcing that "DOS is dead" and stating that version 4 was the last full release. MS-DOS had grown in spurts, with many significant features being taken (or duplicated) from other products and operating systems, as well as incorporating the functionality of tools and utilities developed by independent companies to improve the functionality of MS-DOS, including Norton Utilities, PC Tools (Microsoft Anti-Virus), QEMM expanded memory manager, DOS/4GW (a 32-bit DOS extender), Stacker disk compression, and others. OS/2 was designed for efficient multitaskingan IBM specialty derived from deep experience with mainframe operating systemsand offered a number of advanced features that had been designed together with similar look and feel; it was seen as the legitimate heir to the "kludgy" DOS platform. During the period when Digital Research was competing in the operating system market some computers, like Amstrad PC-1512, were sold with floppy disks for two operating systems (only one of which could be used at a time), MS-DOS and CP-M86 or a derivative of it. Digital Research produced DOS Plus, which was compatible with MS-DOS 2.11, supported CP/M-86 programs, had additional features including multi-tasking, and could read and write disks in CP/M and MS-DOS format. While OS/2 was under protracted development, Digital Research released the MS-DOS compatible DR-DOS 5, which included features only available as third-party add-ons for MS-DOS (and still maintained considerable internal CP/M-86 compatibility). Unwilling to lose any portion of the market, Microsoft responded by announcing the "pending" release of MS-DOS 5.0 in May 1990. This effectively killed most DR-DOS sales until the actual release of MS-DOS 5.0

in June 1991. Digital Research brought out DR-DOS 6, which sold well until the"pre-announcement" of MS-DOS 6.0 again stifled the sales of DR-DOS.

3. The File System

In computing, a **file system** (often also written as **filesystem**) is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. File systems may use a data storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files, they might provide access to data on a file server by acting as clients for a network protocol (e.g., NFS, SMB, or 9P clients), or they may be virtual and exist only as an access method for virtual data (e.g., procfs). It is distinguished from a directory service and registry.

More formally, a file system is a special-purpose database for the storage, organization, manipulation, and retrieval of data.

4. Aspects of file systems

Most file systems make use of an underlying data storage device that offers access to an array of fixed-size blocks, sometimes called sectors, generally a power of 2 in size (512 bytes or 1, 2, or 4 KiB are most common). The file system software is responsible for organizing these sectors into files and directories, and keeping track of which sectors belong to which file and which are not being used. Most file systems address data in fixed-sized units called "clusters" or "blocks" which contain a certain number of disk sectors (usually 1-64). This is the smallest amount of disk space that can be allocated to hold a file.

However, file systems need not make use of a storage device at all. A file system can be used to organize and represent access to any data, whether it be stored or dynamically generated (e.g., procfs).

5. File names

Whether the file system has an underlying storage device or not, file systems typically have directories which associate **file names** with files, usually by connecting the file name to an index in a file allocation table of some sort, such as the FAT in a DOS file system, or an inode in a Unix-like file system. Directory structures may be flat, or allow hierarchies where directories may contain subdirectories. In some file systems, file names are structured, with special syntax for filename extensions and version numbers. In others, file names are simple strings, and per-file metadata is stored elsewhere.

6. Metadata

Other bookkeeping information is typically associated with each file within a file system. The length of the data contained in a file may be stored as the number of blocks allocated for the file or as an exact byte count. The time that the file was last modified may be stored as the file's timestamp. Some file systems also store the file creation time, the time it was last accessed, and the time that the file's meta-data was changed. (Note that many early PC operating systems did not keep track of file times.) Other information can include the file's device type (e.g., block, character, socket, subdirectory, etc.), its owner user-ID and group-ID, and its access permission settings (e.g., whether the file is read-only, executable, etc.). Arbitrary attributes can be associated on advanced file systems, such as XFS, ext2/ext3, some versions of UFS, and HFS+, using extended file attributes. This feature is implemented in the kernels of Linux, FreeBSD and Mac OS X operating systems, and allows metadata to be associated with the file at the file system level. This, for example, could be the author of a document, the character encoding of a plain-text document, or a checksum.

7. Hierarchical file systems

The hierarchical file system was an early research interest of Dennis Ritchie of UNIX fame; previous implementations were restricted to only a few levels, notably the IBM implementations, even of their early databases like IMS. After the success of UNIX, Ritchie extended the file system concept to every object in his later operating system developments, such as Plan 9 and Inferno.

8. Facilities

Traditional file systems offer facilities to create, move and delete both files and directories. They lack facilities to create additional links to a directory (hard links in Unix), rename parent links (".." in Unix-like OS), and create bidirectional links to files. Traditional file systems also offer facilities to truncate, append to, create, move, delete and in-place modify files. They do not offer facilities to prepend to or truncate from the beginning of a file, let alone arbitrary insertion into or deletion from a file. The operations provided are highly asymmetric and lack the generality to be useful in unexpected contexts. For example, interprocess pipes in UNIX have to be implemented outside of the file system because the pipes concept does not offer truncation from the beginning of files.

9. Pipeline

In software engineering, a pipeline consists of a chain of processing elements (processes, threads, coroutines, etc.), arranged so that the output of each element is the input of the next. Usually some amount of buffering is provided between consecutive elements. The information that flows in these pipelines is often a stream of records, bytes or bits.
The concept is also called the pipes and filters design pattern. It was named by analogy to a physical pipeline.

10. Multiprocessed pipelines

Pipelines are often implemented in a multitasking OS, by launching all elements at the same time as processes, and automatically servicing the data read requests by each process with the data written by the upstream process. In this way, the CPU will be naturally switched among the processes by the scheduler so as to minimize its idle time. In other common models elements are implemented as lightweight threads or as co routines, to reduce the OS overhead often involved with processes. Depending upon the OS, threads may be scheduled directly by the OS or by a thread manager. Co routines are always scheduled by a co routine manager of some form. Usually, read and write requests are blocking operations, which means that the execution of the source process, upon writing, is suspended until all data could be written to the destination process, and, likewise, the execution of the destination process, upon reading, is suspended until at least some of the requested data could be obtained from the source process. Obviously, this cannot lead to a deadlock, where both processes would wait indefinitely for each other to respond, since at least one of the two processes will soon thereafter have its request serviced by the operating system, and continue to run. For performance, most operating systems implementing pipes use pipe buffers, which allow the source process to provide more data than the destination process is currently able or willing to receive. Under most Unices and Unix-like operating systems, a special command is also available which implements a pipe buffer of potentially much larger and configurable size, typically called "buffer". This command can be useful if the destination process is significantly slower than the source process, but it is anyway desired that the source process can complete its task as soon as possible. E.g., if the source process consists of a command which reads an audio track from a CD and the destination process consists of a command which compresses the waveform audio data to a format like OGG Vorbis. In this case, buffering the entire track in a pipe buffer would allow the CD drive to spin down more quickly, and enable the user to remove the CD from the drive before the encoding process has

finished. Such a buffer command can be implemented using available operating system primitives for reading and writing data. Wasteful busy waiting can be avoided by using facilities such as poll or select or multithreading.

11. Returning to Windows

Microsoft Windows is a series of software operating systems and graphical user interfaces produced by Microsoft. Microsoft first introduced an operating environment named Windows in November 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market, overtaking Mac OS, which had been introduced previously. At the 2004 IDC Directions conference, it was stated that Windows had approximately 90% of the client operating system market. The most recent client version of Windows is Windows Vista; the most recent server version is Windows Server 2008. Vista's successor, Windows 7 (currently in public beta) is slated to be released in July 2009. The term Windows collectively describes any or all of several generations of Microsoft operating system products. These products are generally categorized as follows:

12. Early versions

The history of Windows dates back to September 1981, when the project named "Interface Manager" was started. It was announced in November 1983 (after the Apple Lisa, but before the Macintosh) under the name "Windows", but Windows 1.0 was not released until November 1985.The shell of Windows 1.0 was a program known as the MS-DOS Executive. Other supplied programs are Calculator, Calendar, Cardfile, Clipboard viewer, Clock, Control Panel, Notepad, Paint, Reversi, Terminal, and Write. Windows 1.0 does not allow overlapping windows, due to Apple Computer owning this feature. Instead all windows are tiled. Only dialog boxes can appear over other windows. Windows 2.0 was released in October 1987 and featured several improvements to the user interface and memory management. Windows 2.0 allowed application windows to overlap each other and also introduced more sophisticated keyboard-shortcuts. It could also make use of expanded memory. Windows 2.1 was released in two different flavors: Windows/386 employed the 386 virtual 8086 mode to multitask several DOS programs, and the paged memory model to emulate expanded memory using available extended memory. Windows/286 (which, despite its name, would run on the 8086) still ran in real mode, but could make use of the high memory area. The early versions of Windows were often thought of as simply graphical user

interfaces, mostly because they ran on top of MS-DOS and used it for file system services. However, even the earliest 16-bit Windows versions already assumed many typical operating system functions; notably, having their own executable file format and providing their own device drivers (timer, graphics, printer, mouse, keyboard and sound) for applications. Unlike MS-DOS, Windows allowed users to execute multiple graphical applications at the same time, through cooperative multitasking. Windows implemented an elaborate, segment-based, software virtual memory scheme, which allowed it to run applications larger than available memory: code segments and resources were swapped in and thrown away when memory became scarce, and data segments moved in memory when a given application had relinquished processor control, typically waiting for user input.

Topic : The Microsoft Windows User Interface

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Windows XP
- Learn about File systems under Microsoft Windows
- Understand theData retrieval process

Definition/Overview:

Windows XP: Windows XP is a line of operating systems produced by Microsoft for use on personal computers, including home and business desktops, notebook computers, and media centers.

File systems under Microsoft Windows: The File Allocation Table (FAT) filing system, supported by all versions of Microsoft Windows, was an evolution of that used in Microsoft's earlier operating system (MS-DOS which in turn was based on 86-DOS).

Data retrieval process: *The operating system calls on the IFS (Installable File System) manager.*

Key Points:

1. Windows XP

Windows XP is a line of operating systems produced by Microsoft for use on personal computers, including home and business desktops, notebook computers, and media centers. The name "XP" is short for "experience". Windows XP is the successor to both Windows 2000 Professional and Windows Me, and is the first consumer-oriented operating system produced by Microsoft to be built on the Windows NT kernel and architecture. Windows XP was first released on 25 October 2001, and over 400 million copies were in use in January 2006, according to an estimate in that month by an IDC analyst. It is succeeded by Windows Vista, which was released to volume license customers on 8 November 2006 and worldwide to the general public on 30 January 2007. Direct OEM and retail sales of Windows XP ceased on 30 June 2008, although it is still possible to obtain Windows XP from System Builders (smaller OEMs who sell assembled computers) until 31 July 2009 or by purchasing Windows Vista Ultimate or Business and then downgrading to Windows XP. The most common editions of the operating system are Windows XP Home Edition, which is targeted at home users, and Windows XP Professional, which offers additional features such as support for Windows Server domains and two physical processors, and is targeted at power users, business and enterprise clients. Windows XP Media Center Edition has additional multimedia features enhancing the ability to record and watch TV shows, view DVD movies, and listen to music. Windows XP Tablet PC Edition is designed to run ink-aware applications built using the Tablet PC platform. Two separate 64-bit versions of Windows XP were also released, Windows XP 64-bit Edition for IA-64 (Itanium) processors and Windows XP Professional x64 Edition for x86-64. There is also Windows XP Embedded, a componentized version of the Windows XP Professional, and editions for specific markets such as Windows XP Starter Edition. Windows XP is known for its improved stability and efficiency over the 9x versions of Microsoft Windows. It presents a significantly redesigned graphical user interface, a change Microsoft promoted as more user-friendly than previous versions of Windows. A new software management facility called Side-by-Side Assembly was introduced to avoid the "DLL hell" that plagued older consumer-oriented 9x versions of Windows. It is also the first version of Windows to use product activation to combat software

piracy, a restriction that did not sit well with some users and privacy advocates. Windows XP has also been criticized by some users for security vulnerabilities, tight integration of applications such as Internet Explorer 6 and Windows Media Player, and for aspects of its default user interface. Later versions with Service Pack 2, and Internet Explorer 7 addressed some of these concerns. During development, the project was codenamed "Whistler", after Whistler, British Columbia, as many Microsoft employees skied at the Whistler-Blackcomb ski resort. As of the end of November 2008, Windows XP is the most widely used operating system in the world with a 66.31% market share, having peaked at 85% in December 2006. New and updated features Windows XP introduced several new features to the Windows line, including:

- Faster start-up and hibernation sequences
- The ability to discard a newer device driver in favor of the previous one (known as driver rollback), should a driver upgrade not produce desirable results
- A new, arguably more user-friendly interface, including the framework for developing themes for the desktop environment
- Fast user switching, which allows a user to save the current state and open applications of their desktop and allow another user to log on without losing that information
- The ClearType font rendering mechanism, which is designed to improve text readability on Liquid Crystal Display (LCD) and similar monitors
- Remote Desktop functionality, which allows users to connect to a computer running Windows XP Pro from across a network or the Internet and access their applications, files, printers, and devices
- Support for most DSL modems and wireless network connections, as well as networking over FireWire, and Bluetooth.

User interface Windows XP features a new task-based graphical user interface. The Start menu and search capability were redesigned and many visual effects were added, including:

- A translucent blue selection rectangle in Explorer
- Drop shadows for icon labels on the desktop
- Task-based sidebars in Explorer windows ("common tasks")
- The ability to group the taskbar buttons of the windows of one application into one button
- The ability to lock the taskbar and other toolbars to prevent accidental changes
- The highlighting of recently added programs on the Start menu

- Shadows under menus (Windows 2000 had shadows under mouse pointers, but not menus)

Windows XP analyzes the performance impact of visual effects and uses this to determine whether to enable them, so as to prevent the new functionality from consuming excessive additional processing overhead. Users can further customize these settings. Some effects, such as alpha blending (transparency and fading), are handled entirely by many newer video cards. However, if the video card is not capable of hardware alpha blending, performance can be substantially hurt, and Microsoft recommends the feature should be turned off manually. Windows XP adds the ability for Windows to use "Visual Styles" to change the user interface. However, visual styles must be cryptographically signed by Microsoft to run. Luna is the name of the new visual style that ships with Windows XP, and is enabled by default for machines with more than 64 MiB of video RAM. Luna refers only to one particular visual style, not to all of the new user interface features of Windows XP as a whole. Some users "patch" the uxtheme.dll file that restricts the ability to use visual styles, created by the general public or the user, on Windows XP. In addition to the included Windows XP themes, there is one previously unreleased theme with a dark blue taskbar and window bars similar to Windows Vista titled "Royale Noir" available for download, albeit unofficially. Microsoft officially released a modified version of this theme as the "Zune" theme, to celebrate the launch of its Zune portable media player in November 2006. The differences are only visual with a new glassy look along with a black taskbar instead of dark blue and an orange start button instead of green. Additionally, the Media Center"Royale" theme, which was included in the Media Centereditions, is also available to download for use on all Windows XP editions. The default wallpaper, Bliss, is a BMP photograph of a landscape in the Napa Valleyoutside Napa, California, with rolling green hills and a blue sky with stratocumulus and cirrus clouds. The Windows 2000 "classic" interface can be used instead if preferred. Several third party utilities exist that provide hundreds of different visual styles. Microsoft licensed technology from WindowBlinds creator Stardock to create its visual styles in XP.

2.File systems under Microsoft Windows

The File Allocation Table (FAT) filing system, supported by all versions of Microsoft Windows, was an evolution of that used in Microsoft's earlier operating system (MS-DOS which in turn was based on 86-DOS). FAT ultimately traces its roots back to the short-lived M-DOS project and Standalone disk BASIC before it. Over the years various features have been added to it, inspired by similar features found on file systems used by operating systems

such as UNIX. Older versions of the FAT file system (FAT12 and FAT16) had file name length limits, a limit on the number of entries in the root directory of the file system and had restrictions on the maximum size of FAT-formatted disks or partitions. Specifically, FAT12 and FAT16 had a limit of 8 characters for the file name, and 3 characters for the extension (such as .exe). This is commonly referred to as the 8.3 filename limit. VFAT, which was an extension to FAT12 and FAT16 introduced in Windows NT 3.5 and subsequently included in Windows 95, allowed long file names (LFN). FAT32 also addressed many of the limits in FAT12 and FAT16, but remains limited compared to NTFS. NTFS, introduced with the Windows NT operating system, allowed ACL-based permission control. Hard links, multiple file streams, attribute indexing, quota tracking, compression and mount-points for other file systems (called "junctions") are also supported, though not all these features are well-documented. Unlike many other operating systems, Windows uses a drive letter abstraction at the user level to distinguish one disk or partition from another. For example, the path C:\WINDOWSrepresents a directory WINDOWSon the partition represented by the letter C. The C drive is most commonly used for the primary hard disk partition, on which Windows is usually installed and from which it boots. This "tradition" has become so firmly ingrained that bugs came about in older versions of Windows which made assumptions that the drive that the operating system was installed on was C. The tradition of using "C" for the drive letter can be traced to MS-DOS, where the letters A and B were reserved for up to two floppy disk drives. Network drives may also be mapped to drive letters.

3. Data retrieval process

The operating system calls on the IFS (Installable File System) manager. The IFS calls on the correct FSD (File System Driver) in order to open the selected file from a choice of four FSDs that work with different storage systemsNTFS, VFAT, CDFS (for optical drives), and Network. The FSD gets the location on the disk for the first cluster of the file from the FAT (File Allocation Table), FAT32, VFAT (Virtual File Allocation Table), or, in the case of Windows NT based, the MFT (Master File Table). In short, the whole point of the FAT, FAT32, VFAT, or MFT is to map out all the files on the disk and record where they are located (which track and sector of the disk).

Topic : The Unix/Linux User Interface

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Unix
- Understand the file system
- Understand the File systems under Linux
- Understand the File systems under Mac OS X
- Understand the Pipes, Filters, and Redirection
- Understand the Pipelines in command line interfaces
- Understand the Shell Scripts
- Understand the Other types of shells
- Understand the Shell categories

Definition/Overview:

UNIX:UNIX (officially trademarked as **UNIX**, sometimes also written as UNIXwith small caps) is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna. Unix-like operating systems create a virtual file system, which makes all the files on all the devices appear to exist in a single hierarchy.

File systems under Linux: *Linux supports many different file systems, but common choices for the system disk include the ext* family (such as ext2 and ext3), XFS, JFS and ReiserFS.*

File systems under Mac OS X: *Mac OS X uses a file system that it inherited from classic Mac OS called HFS Plus. HFS Plus is a metadata-rich and case preserving file system.*

Key Points:

1. Unix

UNIX (officially trademarked as **UNIX**, sometimes also written as UNIXwith small caps) is a computer operating system originally developed in 1969 by a group of AT&T employees at

Bell Labs, including Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna. Today's UNIX systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organizations. As of 2007, the owner of the trademark is The Open Group, an industry standards consortium. Only systems fully compliant with and certified to the Single UNIX Specification are qualified to use the trademark; others are called "UNIXsystem-like" or "Unix-like". During the late 1970s and early 1980s, the influence of UNIXin academic circles led to large-scale adoption of UNIX(particularly of the BSD variant, originating from the University of California, Berkeley) by commercial startups, the most notable of which are Solaris, HP-UX, and AIX. Today, in addition to certified UNIX systems, Unix-like operating systems such as Linux and BSD are commonly encountered. Sometimes, "traditional Unix" may be used to describe a UNIX or an operating system that has the characteristics of either Version 7 UNIX or UNIX System V. UNIX operating systems are widely used in both servers and workstations. The UNIX environment and the client-server program model were essential elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers. Both UNIX and the C programming language were developed by AT&T and distributed to government and academic institutions, causing both to be ported to a wider variety of machine families than any other operating system. As a result, UNIX became synonymous with "open systems". UNIX was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. UNIX systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality.

These concepts are known as the UNIX philosophy. Under UNIX, the "operating system" consists of many of these utilities along with the master control program, the kernel. The kernel provides services to start and stop programs, handle the file system and other common "low level" tasks that most programs share, and, perhaps most importantly, schedules access to hardware to avoid conflicts if two programs try to access the same resource or device simultaneously. To mediate such access, the kernel was given special rights on the system, leading to the division between user-space and kernel-space. The microkernel concept was introduced in an effort to reverse the trend towards larger kernels and return to a system in which most tasks were completed by smaller utilities. In an era when a "normal" computer

consisted of a hard disk for storage and a data terminal for input and output (I/O), the UNIX file model worked quite well as most I/O was "linear". However, modern systems include networking and other new devices. As graphical user interfaces developed, the file model proved inadequate to the task of handling asynchronous events such as those generated by a mouse, and in the 1980s non-blocking I/O and the set of inter-process communication mechanisms was augmented (sockets, shared memory, message queues, semaphores), and functionalities such as network protocols were moved out of the kernel. Beginning in the late 1980s, an open operating system standardization effort now known as POSIX provided a common baseline for all operating systems; IEEE based POSIX around the common structure of the major competing variants of the UNIX system, publishing the first POSIX standard in 1988. In the early 1990s a separate but very similar effort was started by an industry consortium, the Common Open Software Environment (COSE) initiative, which eventually became the Single UNIX Specification administered by The Open Group. Starting in 1998 the Open Group and IEEE started the Austin Group, to provide a common definition of POSIX and the Single UNIX Specification. In an effort towards compatibility, in 1999 several UNIX system vendors agreed on SVR4's Executable and Linkable Format (ELF) as the standard for binary and object code files. The common format allows substantial binary compatibility among UNIX systems operating on the same CPU architecture.

2. The file system

Unix-like operating systems create a virtual file system, which makes all the files on all the devices appear to exist in a single hierarchy. This means, in those systems, there is one root directory, and every file existing on the system is located under it somewhere. Unix-like systems can use a RAM disk or network shared resource as its root directory. Unix-like systems assign a device name to each device, but this is not how the files on that device are accessed. Instead, to gain access to files on another device, the operating system must first be informed where in the directory tree those files should appear. This process is called mounting a file system. For example, to access the files on a CD-ROM, one must tell the operating system "Take the file system from this CD-ROM and make it appear under such-and-such directory". The directory given to the operating system is called the mount point - it might, for example, be /media. The /media directory exists on many UNIX systems (as specified in the Filesystem Hierarchy Standard) and is intended specifically for use as a mount point for removable media such as CDs, DVDs and like floppy disks. It may be empty,

or it may contain subdirectories for mounting individual devices. Generally, only the administrator (i.e. root user) may authorize the mounting of file systems. Unix-like operating systems often include software and tools that assist in the mounting process and provide it new functionality. Some of these strategies have been coined "auto-mounting" as a reflection of their purpose.

- In many situations, file systems other than the root need to be available as soon as the operating system has booted. All Unix-like systems therefore provide a facility for mounting file systems at boot time. System administrators define these file systems in the configuration file fstab or vfstab in Solaris Operating Environment, which also indicates options and mount points.
- In some situations, there is no need to mount certain file systems at boot time, although their use may be desired thereafter. There are some utilities for Unix-like systems that allow the mounting of predefined file systems upon demand.
- Removable media have become very common with microcomputer platforms. They allow programs and data to be transferred between machines without a physical connection. Common examples include USB flash drives, CD-ROMs, and DVDs. Utilities have therefore been developed to detect the presence and availability of a medium and then mount that medium without any user intervention.
- Progressive Unix-like systems have also introduced a concept called supermounting; see, for example, the Linux supermount-ng project. For example, a floppy disk that has been supermounted can be physically removed from the system. Under normal circumstances, the disk should have been synchronized and then unmounted before its removal. Provided synchronization has occurred, a different disk can be inserted into the drive. The system automatically notices that the disk has changed and updates the mount point contents to reflect the new medium. Similar functionality is found on Windows machines.
- A similar innovation preferred by some users is the use of autofs, a system that, like supermounting, eliminates the need for manual mounting commands. The difference from supermount, other than compatibility in an apparent greater range of applications such as access to file systems on network servers, is that devices are mounted transparently when requests to their file systems are made, as would be appropriate for file systems on network servers, rather than relying on events such as the insertion of media, as would be appropriate for removable media.

3. File systems under Linux

Linux supports many different file systems, but common choices for the system disk include the ext* family (such as ext2 and ext3), XFS, JFS and ReiserFS. The Sun Microsystems Solaris operating system in earlier releases defaulted to (non-journaled or non-logging) UFS for bootable and supplementary file systems. Solaris (as most operating systems based upon open standards and/or open source) defaulted to, supported, and extended UFS. Support for other file systems and significant enhancements were added over time, including Veritas Software Corp. (Journaling) VxFS, Sun Microsystems (Clustering) QFS, Sun Microsystems (Journaling) UFS, and Sun Microsystems (open source, poolable, 128 bit compressible, and error-correcting) ZFS. Kernel extensions were added to Solaris to allow for bootable Veritas VxFS operation. Logging or Journaling was added to UFS in Sun's Solaris 7. Releases of Solaris 10, Solaris Express, OpenSolaris, and other open source variants of the Solaris operating system later supported bootable ZFS. Logical Volume Management allows for spanning a file system across multiple devices for the purpose of adding redundancy, capacity, and/or throughput. Legacy environments in Solaris may use Solaris Volume Manager (formerly known as Solstice DiskSuite.) Multiple operating systems (including Solaris) may use Veritas Volume Manager. Modern Solaris based operating systems eclipse the need for Volume Management through leveraging virtual storage pools in ZFS.

4. File systems under Mac OS X

Mac OS X uses a file system that it inherited from classic Mac OS called HFS Plus. HFS Plus is a metadata-rich and case preserving file system. Due to the UNIX roots of Mac OS X, UNIX permissions were added to HFS Plus. Later versions of HFS Plus added journaling to prevent corruption of the file system structure and introduced a number of optimizations to the allocation algorithms in an attempt to defragment files automatically without requiring an external defragmenter. Filenames can be up to 255 characters. HFS Plus uses Unicode to store filenames. On Mac OS X, the filetype can come from the type code, stored in file's metadata, or the filename. HFS Plus has three kinds of links: Unix-style hard links, Unix-style symbolic links and aliases. Aliases are designed to maintain a link to their original file even if they are moved or renamed; they are not interpreted by the file system itself, but by the File Manager code in userland. Mac OS X also supports the UFS file system, derived from the BSD UNIX Fast File System via NeXTSTEP. However, as of Mac OS X 10.5 (Leopard), Mac OS X can no longer be installed on a UFS volume, nor can a pre-Leopard system installed on a UFS volume be upgraded to Leopard. File systems under Plan 9 from

Bell Labs Plan 9 from Bell Labs was originally designed to extend some of UNIXs good points, and to introduce some new ideas of its own while fixing the shortcomings of Unix. With respect to file systems, the UNIX system of treating things as files was continued, but in Plan 9, everything is treated as a file, and accessed as a file would be (i.e., no ioctl or mmap). Perhaps surprisingly, while the file interface is made universal it is also simplified considerably, for example symlinks, hard links and suid are made obsolete, and an atomic create/open operation is introduced. More importantly the set of file operations becomes well defined and subversions of this like ioctl are eliminated.

Secondly, the underlying 9P protocol was used to remove the difference between local and remote files (except for a possible difference in latency). This has the advantage that a device or devices, represented by files, on a remote computer could be used as though it were the local computer's own device(s). This means that under Plan 9, multiple file servers provide access to devices, classing them as file systems. Servers for "synthetic" file systems can also run in user space bringing many of the advantages of micro kernel systems while maintaining the simplicity of the system. Everything on a Plan 9 system has an abstraction as a file; networking, graphics, debugging, authentication, capabilities, encryption, and other services are accessed via I-O operations on file descriptors. For example, this allows the use of the IP stack of a gateway machine without need of NAT, or provides a network-transparent window system without the need of any extra code. Another example: a Plan-9 application receives FTP service by opening an FTP site. The ftpfs server handles the open by essentially mounting the remote FTP site as part of the local file system. With ftpfs as an intermediary, the application can now use the usual file-system operations to access the FTP site as if it were part of the local file system. A further example is the mail system which uses file servers that synthesize virtual files and directories to represent a user mailbox as /mail/fs/mbox. The wikifs provides a file system interface to a wiki. These file systems are organized with the help of private, per-process namespaces, allowing each process to have a different view of the many file systems that provide resources in a distributed system.

5. Pipes, Filters, and Redirection

In Unix-like computer operating systems, a pipeline is the original software pipeline: a set of processes chained by their standard streams, so that the output of each process (stdout) feeds directly as input (stdin) of the next one. Each connection is implemented by an anonymous pipe. Filter programs are often used in this configuration. The concept was invented by

Douglas McIlroy for UNIX shells and it was named by analogy to a physical pipeline. In this example, ls is the UNIX directory lister, and lessis an interactive text pager with searching capabilities. The pipeline lets the user scroll up and down a directory listing that may not fit on the screen. Pipelines ending in less (or more, a similar text pager) are among the most commonly used. They let the user navigate potentially large amounts of text (constrained only by available memory) which otherwise would have scrolled past the top of the terminal and been lost. Put differently, they relieve programmers from the burden of implementing text pagers in their applications: they can pipe output through less, or assume that the user will do so when needed. Below is an example of a pipeline that implements a kind of spell checker for the web resource indicated by a URL. An explanation of what it does follows.

```
curl "http://en.wikipedia.org/wiki/Pipeline_(Unix)" | \
sed 's/[^a-zA-Z ]/ /g' | \
tr 'A-Z ' 'a-z\n' | \
grep '[a-z]' | \
sort -u | \
comm -23 - /usr/share/dict/words
```

- Note: The character "\" is used to place all six lines into a single command line.
- First, curl obtains the HTML contents of a web page (could use wget on some systems).
- Second, sed removes all characters that are not spaces or letters from the web page's content, replacing them with spaces.
- Third, tr changes all of the uppercase letters into lowercase and converts the spaces in the lines of text to newlines (each 'word' is now on a separate line).
- Fourth, grep includes only lines that contain at least one lowercase alphabetical character (removing any blank lines).
- Fifth, sort sorts the list of 'words' into alphabetical order, and the -u switch removes duplicates.
- Finally, comm finds lines in common between two files, -23 suppresses lines unique to the second file, and those that are common to both, leaving only those that are found only in the first file named. The - in place of a filename causes comm to use its standard input (from the pipe line in this case). This results in a list of "words" (lines) that are not found in /usr/share/dict/words.

- The special character "|" tells the operating system to pipe the output from the previous command in the line into the next command in the line. That is, the output of the curl command is given as the input of the sed command.

6. Pipelines in command line interfaces

Most UNIX shells have a special syntax construct for the creation of pipelines. Typically, one simply writes the filter commands in sequence, separated by the ASCII vertical bar character "|" (which, for this reason, is often called "pipe character" by UNIX users). The shell starts the processes and arranges for the necessary connections between their standard streams (including some amount of buffer storage).

7. Shell Scripts

A**UNIX shell**, is a command interpreter (see shell) and script host that provides a traditional user interface for the UNIX operating system and for Unix-like systems. Users direct the operation of the computer by entering command input as text for a command line interpreter to execute or by creating text scripts of one or more such commands. The most generic sense of the term shell means any program that users use to type commands. Since in the UNIX operating system users can select which shell they want to use (which program should execute when they log in), many shells have been developed. It is called a "shell" because it hides the details of the underlying operating system behind the shell's interface (in contrast with the "kernel", which refers to the lowest-level or 'inner-most' component of an operating system). Similarly, graphical user interfaces for UNIX, such as GNOME, KDE, and Xfce can be called visual shells or graphical shells. By itself, the term shell is usually associated with the command line. In UNIX, any program can be the user's shell. Users who want to use a different syntax for typing commands can specify a different program as their shell, though in practice this usually requires administrator rights. The term shell also refers to a particular program, such as the Bourne shell, sh. The Bourne shell was the shell used in early versions of UNIX and became a de facto standard; every Unix-like system has at least one shell compatible with the Bourne shell. The Bourne shell program is located in the UNIX file hierarchy at /bin/sh. On some systems, such as BSD, /bin/shis a Bourne shell or equivalent, but on other systems such as Linux, /bin/sh is likely to be a link to a compatible, but more feature-rich shell. POSIX specifies its standard shell as a strict subset of the Korn shell.

8. Other types of shells

The UNIX shell was unusual when it was first created. Since it is both an interactive command language and the language used to script the system, it is a scripting programming language. Many shells created for other operating systems offer rough equivalents to UNIX shell functionality. On systems using a windowing system, some users may never use the shell directly. On UNIX systems, the shell is still the implementation language of system startup scripts, including the program that starts the windowing system, the programs that facilitate access to the Internet, and many other essential functions. On MS-DOS, OS/2, and Windows, equivalents to UNIX system scripts are called batch files, and have either a ".bat" or ".cmd" extension. A newer CLI - Windows PowerShell, will replace the existing NT command line, cmd.exe; it has many features derived from UNIX shells, though it uses a somewhat different syntax. Many users of a UNIX system still find a modern command line shell much more convenient for many tasks than any GUI application. Due to the recent movement in favor of open source, most UNIX shells have at least one version that is open source.

9. Shell categories

UNIX shells can be broadly divided into four categories: Bourne-like, C Shell-like, nontraditional, and historical.

In Section 3 of this course you will cover these topics:

- The Intel Architecture
- Ms-Dos Internals
- Windows Xp Internals
- Unixand Linux Internals

Topic : The Intel Architecture

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Intel architecture
- Understand the MCS-4 Family
- Understand the MCS-40 Family:
- Understand the 8008
- Understand the 8080
- Understand the 8085
- Understand the MCS-85 Family:

Definition/Overview:

Intel architecture: This generational and chronological list of Intel microprocessors attempts to present all of Intel's processors from the pioneering 4-bit 4004 (1971) to the present high-end offerings, the 64-bit Itanium 2 (2002) and Intel Core 2 and Xeon 5100 and 7100 series processors (2006).

Memory: Intel Turbo Memory (codenamed Robson, also known as a Robson cache) is a technology introduced by semi-conductor company Intel to utilize NAND flash memory modules, reducing the time it takes for a computer to power up, access programs, and write data to the hard drive.

Memory protection: Memory protection is a way to control memory usage on a computer, and is core to virtually every modern operating system.

Segmentation: The x86 architecture has multiple segmentation features, which are useful for those who wish to use protected memory on this architecture.

Key Points:

1. Intel architecture

This generational and chronological **list of Intel microprocessors** attempts to present all of Intel's processors from the pioneering 4-bit 4004 (1971) to the present high-end offerings, the 64-bit Itanium 2 (2002) and Intel Core 2 and Xeon 5100 and 7100 series processors (2006). Concise technical data are given for each product.

- Introduced November 15, 1971

- Clock rate 740 kHz
- 0.07 MIPS
- Bus Width 4 bits (multiplexed address/data due to limited pins)
- PMOS
- Number of Transistors 2,300 at 10 m
- Addressable Memory 640 bytes
- Program Memory 4 KB (4 KB)
- One of the earliest Commercial Microprocessors (cf. Four Phase Systems AL1, F14 CADC)
- Originally designed to be used in Busicom calculator

2. MCS-4 Family:

- 4004-CPU
- 4001-ROM & 4 Bit Port
- 4002-RAM & 4 Bit Port
- 4003-10 Bit Shift Register
- 4008-Memory+I/O Interface
- 4009-Memory+I/O Interface

3. MCS-40 Family:

- 4040-CPU
- 4101-1024-bit (256 x 4) Static RAM with separate I/O
- 4201-4MHz Clock Generator
- 4207-General Purpose Byte I/O Port
- 4209-General Purpose Byte I/O Port
- 4211-General Purpose Byte I/O Port
- 4265-Programmable General Purpose I/O Device
- 4269-Programmable Keyboard Display Device
- 4289-Standard Memory Interface for MCS-4/40
- 4308-8192-bit (1024 x 8) ROM w/ 4-bit I/O Ports
- 4316-16384-bit (2048 x 8) Static ROM
- 4702-2048-bit (256 x 8) EPROM
- 4801-5.185 MHz Clock Generator Crystal for 4004/4201A or 4040/4201A

4. 8008

- Introduced April 1, 1972
- Clock rate 500 kHz (8008-1: 800 kHz)
- 0.05 MIPS
- Bus Width 8 bits (multiplexed address/data due to limited pins)
- Enhancement load PMOS logic
- Number of Transistors 3,500 at 10 m
- Addressable memory 16 KB
- Typical in dumb terminals, general calculators, bottling machines
- Developed in tandem with 4004
- Originally intended for use in the Datapoint 2200 terminal

5. 8080

- Introduced April 1, 1974
- Clock rate 2 MHz
- 0.64 MIPS
- Bus Width 8 bits data, 16 bits address
- Enhancement load NMOS logic
- Number of Transistors 6,000
- Assembly language downwards compatible with 8008.
- Addressable memory 64 KB
- Up to 10X the performance of the 8008
- Used in the Altair 8800, Traffic light controller, cruise missile
- Required six support chips versus 20 for the 8008

6. 8085

- Introduced March 1976
- Clock rate 5 MHz
- 0.37 MIPS
- Bus Width 8 bits data, 16 bits address
- Depletion load NMOS logic
- Number of Transistors 6,500 at 3 m

- Binary compatible downwards with the 8080.
- Used in Toledo scale. Also was used as a computer peripheral controller - modems, harddisks, etc...
- CMOS 80C85 in Mars Sojourner, Radio Shack Model 100 portable.
- High level of integration, operating for the first time on a single 5 volt power supply, from 12 volts previously. Also featured two serial I/O connection,3 maskable interrupts,1 Non-maskable,1 programmable,status,DMA.

8. MCS-85 Family:

- 8085-CPU
- 8155-RAM+ 3 I/O Ports+Timer "Active Low CS"
- 8156-RAM+ 3 I/O Ports+Timer "Active High CS"
- 8185-SRAM
- 8202-Dynamic RAM Controller]
- 8203-Dynamic RAM Controller
- 8205-1 Of 8 Binary Decoder
- 8206-Error Detection & Correction Unit
- 8207-DRAM Controller
- 8210-TTL To MOS Shifter & High Voltage Clock Driver
- 8212-8 BitI/O Port
- 8216-4 Bit Parallel Bidirectional Bus Driver
- 8219-Bus Controller
- 8222-Dynamic RAM Refresh Controller
- 8226-4 Bit Parallel Bidirectional Bus Driver
- 8231-Arithmetic Processing Unit
- 8232-Floating Point Processor
- 8237-DMA Controller
- 8251-Communication Controller
- 8253-Programmable Interval Timer
- 8254-Programmable Interval Timer
- 8255-Programmable Peripheral Interface
- 8256-Multifunction Support Controller
- 8257-DMA Controller

- 8259-Programmable Interrupt Controller
- 8271-Programmable Floppy Disk Controller
- 8272-Single/Double Density Floppy Disk Controller
- 8273-Programmable HDLC/SDLC Protocol Controller
- 8274-Multi-Protocol Serial Controller
- 8275-CRT Controller
- 8276-Small System CRT Controller
- 8278-Programmable KeyBoard Interface
- 8279-KeyBoard/Display Controller
- 8282-8-bit Non-Inverting Latch with Output Buffer
- 8283-8-bit Inverting Latch with Output Buffer
- 8291-GPIB Talker/Listener
- 8292-GPIB Controller
- 8293-GPIB Transceiver
- 8294-Data Encryption/Decryption Unit+1 O/P Port
- 8295-Dot Matrix Printer Controller
- 8296-GPIB Transceiver
- 8297-GPIB Transceiver
- 8355-16,384-bit (2048 x 8) ROM with I/O
- 8604-4096-bit (512 x 8) PROM
- 8702-2K-bit (256 x 8) PROM
- 8755-EPROM+2 I/O Ports

2. Memory

Intel Turbo Memory (codenamed Robson, also known as a Robson cache) is a technology introduced by semi-conductor company Intel to utilize NAND flash memory modules, reducing the time it takes for a computer to power up, access programs, and write data to the hard drive. The technology was publicly introduced on October 24, 2005 at the Intel Developer Forum (IDF) in Taiwan when it gave a demonstration using a laptop that booted up almost immediately. The technology attempts to decrease hard drive usage by moving frequently accessed data over to the flash memory. Flash memory reacts faster than hard drives and requires less power, allowing notebooks to be faster and more power efficient. The Robson cache connects via a mini-PCIe card with on-board NAND flash memory modules,

supporting new features available in Microsoft Windows Vista, namely ReadyBoost (a hard-drive caching solution via USB flash drives) and ReadyDrive (a hard-drive caching solution via hybrid drives), allowing both read caching and write caching of data. Often this is implemented with a Disk Filtering Option ROM (DFOROM). Intel introduced Intel Turbo Memory on May 9, 2007, on the Santa Rosa platform and their Crestline (GM965) chipsets. Intel introduced Intel Turbo Memory 2.0 on July 15, 2008, on the Montevina platform and their Cantiga (GM47) chipsets. This feature is currently available on several Alienware, Dell, Lenovo, ASUS, Sager and Acer Inc. notebooks.

3. Memory protection

Memory protection is a way to control memory usage on a computer, and is core to virtually every modern operating system. The main purpose of memory protection is to prevent a process running on an operating system from accessing memory beyond that allocated to it. This prevents a bug within the process from affecting other processes, and also prevents malicious software from gaining unauthorized access to the system.

4. Segmentation

The x86 architecture has multiple segmentation features, which are useful for those who wish to use protected memory on this architecture. On the x86 architecture, the Global Descriptor Table and Local Descriptor Tables can be used to reference segments in the computer's memory. Pointers to memory segments on x86 processors can also be stored in the processor's segment registers. Initially x86 processors had 4 segment registers, CS (code segment), SS (stack segment), DS (data segment) and ES (extra segment); later another two segment registers were added FS and GS.

5. Paging

In paging, the memory address space is divided into equal, small pieces, called pages. Using a virtual memory mechanism, each page can be made to reside in any location of the physical memory, or be flagged as being protected. Virtual memory makes it possible to have a linear virtual memory address space and to use it to access blocks fragmented over physical memory address space. Most computer architectures based on pages, most notably x86 architecture, also use pages for memory protection. A page table is used for mapping virtual memory to physical memory. The page table is usually invisible to the process. Page tables make it easier to allocate new memory, as each new page can be allocated from anywhere in physical memory. By such design, it is impossible for an application to access a page that has

not been explicitly allocated to it, simply because any memory address, even a completely random one, that application may decide to use, either points to an allocated page, or generates a page fault (PF) error. Unallocated pages simply do not have any addresses from the application point of view. As a side note, a PF may not be a fatal one. A PF is used not only for memory protection, but also in another interesting way: the OS may intercept PF, and may load a page that has been previously swapped out to disk, and return to the application that caused page fault. This way, the application receives the memory page as needed. This scheme, known as swapping, allows in-memory data not currently in use to be moved to disk storage and back, in a way "invisible" for applications, to increase overall memory capacity.

6. Protection keys

A protection key mechanism divides physical memory up into blocks of a particular size (e.g. 2KB), each of which has an associated numerical value called a protection key. Each process also has a protection key value associated with it. On a memory access the hardware checks that the current process's protection key matches the value associated with the memory block being accessed; if not, an exception occurs. This mechanism was used in the System/360 architecture. The System/360 protection keys described above are associated with physical addresses. They should not be confused with the protection key mechanism used by processors such as the Intel Itanium and the HP Precision Architecture (HP/PA, also known as PA-RISC), which are associated with virtual addresses, and which allow multiple keys per process. In the Itanium and PA computer architectures, translations (TLB entries) have "keys" (Itanium) or "access ids" (PA) associated with them. A running process has several protection key registers - 16 for Itanium, 4 for HP PA. A translation selected by the virtual address has its key compared to each of the protection key registers. If any of them match (plus other possible checks), the access is permitted. If none match, a fault or exception is generated. The software fault handler can, if desired, check the missing key against a larger list of keys maintained by software; thus, the protection key registers inside the processor may be treated as software managed cache of a larger list of keys associated with a process. PA has 15-18 bits of key; Itanium mandates at least 18. Keys are usually associated with "protection domains", such as libraries, modules, etc.

7. Simulated segmentation

Simulation is use of a monitoring program to interpret the machine code instructions of some computer. Such an Instruction Set Simulator can provide memory protection by using a segmentation-like scheme and validating the target address and length of each instruction in

real time before actually executing them. The simulator must calculate the target address and length and compare this against a list of valid address ranges that it holds concerning the thread's environment, such as any dynamic memory blocks acquired since the thread's inception plus any valid shared static memory slots. The meaning of "valid" may change throughout the thread's life depending upon context: it may sometimes be allowed to alter a static block of storage, and sometimes not, depending upon the current mode of execution which may or may not depend on a storage key or supervisor state. It is generally not advisable to use this method of memory protection where adequate facilities exist on a CPU, as this takes valuable processing power from the computer. However it is generally used for debugging and testing purposes to provide an extra fine level of granularity to otherwise generic storage violations and can indicate precisely which instruction is attempting to overwrite the particular section of storage which may have the same storage key as unprotected storage. Early IBM teleprocessing systems such as CICS, for example, multi-threaded commercial transactions in shared and unprotected storage for around 20 years.

8. Improving the Performance of the Intel Architecture

The**Core 2** brand refers to a range of Intel's consumer 64-bit single- and dual-core and 2x2 MCM (Multi-Chip Module) quad-core CPUs with the x86-64 instruction set, based on the Intel Core microarchitecture, derived from the 32-bit dual-core Yonah laptop processor. (Note: The Yonah's silicon chip or die comprised two interconnected cores, each similar to those branded Pentium M). The 2x2 MCM dual-die quad-core CPU had two separate dual-core dies (CPUs)next to each otherin one quad-core MCM package. The Core 2 relegated the Pentium brand to a mid-end market, and reunified laptop and desktop CPU lines, which previously had been divided into the Pentium 4, D, and M brands. The Core microarchitecture returned to lower clock rate and improved processors' usage of both available clock cycles and power compared with preceding NetBurst of the Pentium 4/D-branded CPUs. Core microarchitecture provides more efficient decoding stages, execution units, caches, and buses, reducing the power consumption of Core 2-branded CPUs, while increasing their processing capacity. Intel's CPUs have vary wildly in power consumption according to clock rate, architecture and semiconductor process, shown in the CPU power dissipation tables. The Core 2 brand was introduced on July 27, 2006, comprising the Solo (single-core), Duo (dual-core), Quad (quad-core), and in 2007, the Extreme (dual- or quad-core CPUs for enthusiasts) version. Intel Core 2 processors with vPro technology (designed

for businesses) include the dual-core and quad-core branches. The brand became immediately successful. The processors were introduced into Apple's popular MacBook series of notebooks, at the time Apple CEO Steve Jobs justified the entire switch to Intel from IBM's PowerPC processors by the Core 2 series' ability to provide high performance at low power consumption. The series of processors reasserted Intel's role in the processor market after a period in which AMD processors began significantly encroaching on Intel's market share. The processor series became so successful that AnandTech Editor Anand Lal Shimpi coined the phrase "Conroe" as a verb to describe the releasing of a product that eclipses the competition in a previously hotly contested market.

9. Intel's 64-bit Itanium Architecture

Itanium is the brand name for 64-bit Intel microprocessors that implement the Intel Itanium architecture (formerly called IA-64). Intel has released two processor families using the brand: the original Itanium and the Itanium 2. Starting November 1, 2007, new members of the second family are again called Itanium. The processors are marketed for use in enterprise servers and high-performance computing systems. The architecture originated at Hewlett-Packard (HP) and was later developed by HP and Intel together. Itanium's architecture differs dramatically from the x86 architectures (and the x86-64 extensions) used in other Intel processors. The architecture is based on explicit instruction-level parallelism, in which the compiler makes the decisions about which instructions to execute in parallel. By contrast, other superscalar architectures depend on elaborate processor circuitry to keep track of instruction dependencies during runtime. This alternative approach helps current Itanium processors execute up to six instructions per clock cycle. After a protracted development process, the first Itanium processor, codenamed Merced, was released in 2001, and more powerful Itanium processors have been released periodically. HP produces most Itanium-based systems, but several other manufacturers have also developed systems based on Itanium. As of 2007, Itanium is the fourth-most deployed microprocessor architecture for enterprise-class systems, behind x86-64, IBM POWER, and SPARC. Intel released its newest Itanium, codenamed Montvale, in November 2007, Intel announced plans to release a quad-core Itanium processor (code-named Tukwila) to server OEMs in late 2008, and subsequently announced a delay until mid 2009, more than two years later than Intel's initial projection. Industry pundit John C. Dvorak, commenting in 2009 on the history of the Itanium processor, said "This continues to be one of the great fiascos of the last 50 years." Tech columnest

Ashlee Vance comments that the delays and underperformance "turned the product into a joke in the chip industry."

10. Original Itanium processor: 200102

By the time Itanium was released in June, 2001, it was no longer superior to contemporaneous RISC and CISC processors. Itanium competed at the low-end (primarily 4-CPU and smaller systems) with servers based on x86 processors, and at the high end with IBM's POWER architecture and Sun Microsystems' SPARC architecture. Intel repositioned Itanium to focus on high-end business and HPC computing, attempting to duplicate x86's successful "horizontal" market (i.e., single architecture, multiple systems vendors). The success of this initial processor version was limited to replacing PA-RISC and Alpha in HP systems and MIPS in SGI's HPC systems, though IBM also delivered a supercomputer based on this processor. POWER and SPARC remained strong, while the 32-bit x86 architecture continued to grow into the enterprise space. With economies of scale fueled by its enormous installed base, x86 has remained the preeminent "horizontal" architecture in enterprise computing. Only a few thousand systems using the original Itanium processor were sold, due to relatively poor performance, high cost and limited software availability. Recognizing that the lack of software could be a serious issue moving forward, Intel made thousands of these early systems available to independent software vendors (ISVs) to stimulate development. HP and Intel brought the next-generation Itanium 2 processor to market a year later.

11. Architecture

Intel has extensively documented the Itanium instruction set and micro architecture, and the technical press has provided overviews. The architecture has been renamed several times during its history. HP originally called it PA-WideWord. Intel later called it IA-64, then Itanium Processor Architecture (IPA), before settling on Intel Itanium Architecture, but it is still widely referred to as IA-64. It is 64-bit register-rich explicitly-parallel architecture. The base data word is 64 bits, byte-addressable. The logical address space is 2^{64} bytes. The architecture implements predication, speculation, and branch prediction. It uses a hardware register renaming mechanism rather than simple register windowing for parameter passing. The same mechanism is also used to permit parallel execution of loops. Speculation, prediction, predication, and renaming are under control of the compiler: each instruction word includes extra bits for this. This approach is the distinguishing characteristic of the architecture. The architecture implements 128 integer registers, 128 floating point registers, 64 one-bit predicates, and eight branch registers. The floating point registers are 82 bits long to preserve precision for intermediate results. Instruction execution Each 128-bit instruction

word contains three instructions, and the fetch mechanism can read up to two instruction words per clock from the L1 cache into the pipeline. When the compiler can take maximum advantage of this, the processor can execute six instructions per clock cycle. The processor has thirty functional execution units in eleven groups. Each unit can execute a particular subset of the instruction set, and each unit executes at a rate of one instruction per cycle unless execution stalls waiting for data. While not all units in a group execute identical subsets of the instruction set, common instructions can be executed in multiple units.
The execution unit groups include:

- Six general-purpose ALUs, two integer units, one shift unit
- Four data cache units
- Six multimedia units, two parallel shift units, one parallel multiply, one population count
- Two floating-point multiply-accumulate units, two "miscellaneous" floating-point units
- Three branch units

The compiler can often group instructions into sets of six that can execute at the same time. Since the floating-point units implement a multiply-accumulate operation, a single floating point instruction can perform the work of two instructions when the application requires a multiply followed by an add: this is very common in scientific processing. When it occurs, the processor can execute four FLOPs per cycle. For example, the 800 MHz Itanium had a theoretical rating of 3.2 GFLOPS and the fastest Itanium 2, at 1.67 GHz, was rated at 6.67 GFLOPS.

12. Memory architecture

From 2002 to 2006, Itanium 2 processors shared a common cache hierarchy. They had 16 KB of Level 1 instruction cache and 16 KB of Level 1 data cache. The L2 cache was unified (both instruction and data) and is 256 KB. The Level 3 cache was also unified and varied in size from 1.5 MB to 24 MB. The 256 KB L2 cache contains sufficient logic to handle semaphore operations without disturbing the main arithmetic logic unit (ALU). Main memory is accessed through a bus to an off-chip chipset. The Itanium 2 bus was initially called the McKinley bus, but is now usually referred to as the Itanium bus. The speed of the bus has increased steadily with new processor releases. The bus transfers 2x128 bits per clock cycle, so the 200 MHz McKinley bus transferred 6.4 GB/s and the 533 MHz Montecito bus transfers 17.056 GB/s.

13. Architectural changes

Itanium processors released prior to 2006 had hardware support for the IA-32 architecture to permit support for legacy server applications, but performance for IA-32 code was much worse than for native code and also worse than the performance of contemporaneous x86 processors. In 2005, Intel developed the IA-32 Execution Layer (IA-32 EL), a software emulator that provides better performance. With Montecito, Intel therefore eliminated hardware support for IA-32 code.

In 2006, with the release of Montecito, Intel made a number of enhancements to the basic processor architecture including:

- Hardware Multithreading: Each processor core maintains context for two threads of execution. When one thread stalls during memory access, the other thread can execute. Intel calls this "coarse multithreading" to distinguish it from the "hyperthreading technology" Intel integrated into some x86 and x86-64 microprocessors. Coarse multithreading is well matched to the Intel Itanium Architecture and results in an appreciable performance gain.
- Hardware Support for Virtualization: Intel added Intel Virtualization Technology (IntelVT), which provides hardware assists for core virtualization functions. Virtualization allows a software "hypervisor" to run multiple operating system instances on the processor concurrently.
- Cache Enhancements: Montecito added a split L2 cache, which included a dedicated 1 MB L2 cache for instructions. The original 256 KB L2 cache was converted to a dedicated data cache. Montecito also included up to 12MB of on-die L3 cache.

Topic : Ms-Dos Internals

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the MS-DOS
- Understand the End of MS-DOS

Definition/Overview:

MS-DOS: MS-DOS (short for **M**icro**s**oft **D**isk **O**perating**S**ystem) is an operating system commercialized by Microsoft. It was the most commonly used member of the DOS family of operating systems and was the main operating system for personal computers during the 1980s. It was based on the Intel 8086 family of microprocessors, particularly the IBM PC and compatibles.

End of MS-DOS: MS-DOS has effectively ceased to exist as a platform for desktop computing. Since the releases of Windows 9x, it was integrated as a full product mostly used for bootstrapping, and no longer officially released as a standalone DOS, although at first DOS 7 (which was the DOS part included in Windows 95) had been developed as a standalone OS. It was still available, but became increasingly irrelevant as development shifted to the Windows API.

Key Points:

1. MS-DOS

MS-DOS(short for **M**icro**s**oft **D**isk **O**perating System) is an operating system commercialized by Microsoft. It was the most commonly used member of the DOS family of operating systems and was the main operating system for personal computers during the 1980s. It was based on the Intel 8086 family of microprocessors, particularly the IBM PC and compatibles. It was gradually replaced on consumer desktop computers by operating systems offering a graphical user interface (GUI), in particular by various generations of the Microsoft Windows operating system and Linux. MS-DOS was known before as **QDOS**(**Q**uick and **D**irty **O**perating **S**ystem) and 86-DOS. MS-DOS development originally started in 1981, and was first released in 1982 as MS-DOS 1.0. Several versions were released under different names for different hardware. MS-DOS had eight major versions released before Microsoft stopped development in 2000. It was the key product in Microsoft's growth from a programming languages company to a diverse software development firm, providing the company with essential revenue and marketing resources. It was also the underlying basic operating system on which early versions of Windows ran as a GUI.
On microcomputers based on the Intel 8086 and 8088 processorsincluding but by no means restricted to the IBM PC (and clones) architecturethe initial competition to the PC DOS/MS-DOS line came from Digital Research, whose CP/M operating system had inspired MS-DOS.

Digital Research released CP/M-86 a few months after MS-DOS, and it was offered as an alternative to MS-DOS and Microsoft's licensing requirements, but at a higher price. Executable programs for CP/M-86 and MS-DOS were not interchangeable with each other; much applications software was sold in both MS-DOS and CP/M-86 versions until MS-DOS became preponderant (later Digital Research operating systems could run both MS-DOS and CP/M-86 software). MS-DOS supported the simple .COM and the more advanced relocatable .EXE executable file formats; CP/M-86 a relocatable format using the file extension .CMD.
In the later days of MS-DOS, once the IBM-compatible platform was chosen, one would buy any PC clone, get any copy of MS-DOS (or IBM-branded MS-DOS, i.e., PC DOS)done. Many people do not realize that in the early days one chose an IBM PC, a Sirius, a Kaypro, or an Apricot, or other make; these machines all had different architecture and did not even accept the same expansion cards; many of them were not limited to a maximum of 640 kilobytes of system memory, unlike the PC and clones. Then the decision whether to use MS-DOS or CP/M-86 had to be taken, and the appropriate one had to be acquired from the computer manufacturer; it was not possible to use versions of MS-DOS, or PC DOS, interchangeably. One was then tied to using software for the operating system chosen (or, funds permitting, having floppy disks for both, and booting the appropriate one). In the business world the 808x-based machines that MS-DOS was tied to face competition from the UNIX operating system which ran on many different hardware architectures.
Microsoft itself sold a version of UNIX for the PC called Xenix. In the emerging world of home users, a variety of other computers based on various other processors were in serious competition with the IBM PC: the Apple II, early Apple Macintosh, the Commodore 64 and others did not use the 808x processor; many 808x machines of different architectures used custom versions of MS-DOS. At first all these machines were in competition. In time the IBM PC hardware configuration became dominant in the 808x market as software written to communicate directly with the PC hardware without using standard operating system calls ran much faster, but on true PC-compatibles only. Non-PC-compatible 808x machines were too small a market to have fast software written for them alone, and the market remained open only for IBM PCs and machines that closely imitated their architecture, all running either a single version of MS-DOS compatible only with PCs, or the equivalent IBM PC DOS. Most clones cost much less than IBM-branded machines of similar performance, and became widely used by home users, while IBM PCs had a large share of the business computer market. Microsoft and IBM together began what was intended as the follow-on to MS/PC DOS, called OS/2. When OS/2 was released in 1987, Microsoft began an advertising

campaign announcing that "DOS is Dead" and stating that version 4 was the last full release. MS-DOS had grown in spurts, with many significant features being taken (or duplicated) from other products and operating systems, as well as incorporating the functionality of tools and utilities developed by independent companies to improve the functionality of MS-DOS, including Norton Utilities, PC Tools (Microsoft Anti-Virus), QEMM expanded memory manager, DOS/4GW (a 32-bit DOS extender), Stacker disk compression, and others. OS/2 was designed for efficient multitaskingan IBM specialty derived from deep experience with mainframe operating systemsand offered a number of advanced features that had been designed together with similar look and feel; it was seen as the legitimate heir to the "kludgy" DOS platform. During the period when Digital Research was competing in the operating system market some computers, like Amstrad PC-1512, were sold with floppy disks for two operating systems (only one of which could be used at a time), MS-DOS and CP-M86 or a derivative of it. Digital Research produced DOS Plus, which was compatible with MS-DOS 2.11, supported CP/M-86 programs, had additional features including multi-tasking, and could read and write disks in CP/M and MS-DOS format. While OS/2 was under protracted development, Digital Research released the MS-DOS compatible DR-DOS 5, which included features only available as third-party add-ons for MS-DOS (and still maintained considerable internal CP/M-86 compatibility). Unwilling to lose any portion of the market, Microsoft responded by announcing the "pending" release of MS-DOS 5.0 in May 1990.
This effectively killed most DR-DOS sales until the actual release of MS-DOS 5.0 in June 1991. Digital Research brought out DR-DOS 6, which sold well until the "pre-announcement" of MS-DOS 6.0 again stifled the sales of DR-DOS. Microsoft has been accused of carefully orchestrating leaks about future versions of MS-DOS in an attempt to create what in the industry is called FUD (fear, uncertainty, and doubt) regarding DR-DOS. For example, in October 1990, shortly after the release of DR-DOS 5.0, and long before the eventual June 1991 release of MS-DOS 5.0, stories on feature enhancements in MS-DOS started to appear in InfoWorld and PC Week. Brad Silverberg, Vice President of Systems Software at Microsoft and General Manager of its Windows and MS-DOS Business Unit, wrote a forceful letter to PC Week (November 5, 1990), denying that Microsoft was engaged in FUD tactics ("to serve our customers better, we decided to be more forthcoming about version 5.0") and denying that Microsoft copied features from DR-DOS: "The feature enhancements of MS-DOS version 5.0 were decided and development was begun long before we heard about DR-DOS 5.0. There will be some similar features. With 50 million MS-DOS users, it shouldn't be surprising that DRI has heard some of the same requests from customers

that we have." (Schulman et al. 1994). The pact between Microsoft and IBM to promote OS/2 began to fall apart in 1990 when Windows 3.0 became a marketplace success. Much of Microsoft's further contributions to OS/2 also went in to creating a third GUI replacement for DOS, Windows NT. IBM, which had already been developing the next version of OS/2, carried on development of the platform without Microsoft and sold it as the alternative to DOS and Windows.

2. End of MS-DOS

MS-DOS has effectively ceased to exist as a platform for desktop computing. Since the releases of Windows 9x, it was integrated as a full product mostly used for bootstrapping, and no longer officially released as a standalone DOS, although at first DOS 7 (which was the DOS part included in Windows 95) had been developed as a standalone OS. It was still available, but became increasingly irrelevant as development shifted to the Windows API. Windows XP contains a copy of the core MS-DOS 8 files from Windows Millennium, accessible only by formatting a floppy as an "MS-DOS startup disk". Attempting to run COMMAND.COMfrom such a disk under the NTVDM results in the message "Incorrect MS-DOS version". (Note that the DOS boot disk created by Windows XP is even more stripped-down than that created in Windows 98, as it does not include CD-ROM support.) With Windows Vista the files on the startup disk are dated 18th April 2005 but are otherwise unchanged, including the string "MS-DOS Version 8 (C) Copyright 1981-1999 Microsoft Corp" inside COMMAND.COM. However the only versions of DOS currently recognized as stand-alone OSs, and supported as such by the Microsoft Corporation are DOS 6.0 and 6.22, both of which remain available for download via their MSDN, volume license, and OEM license partner websites, for customers with valid login credentials. Today, DOS is still used in embedded x86 systems due to its simple architecture, and minimal memory and processor requirements. The command line interpreter of NT-based versions of Windows, cmd.exe, maintains most of the same commands and some compatibility with DOS batch files.

Topic : Windows Xp Internals

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the Windows XP
- Understand the Windows XP Architecture
- Understand the Process Management.
- Understand the Memory Management
- Understand the Disk Management
- Understand the File Management

Definition/Overview:

Windows XP: Windows XP is a line of operating systems produced by Microsoft for use on personal computers, including home and business desktops, notebook computers, and media centers.

Windows XP Architecture: A main design goal of NT was hardware and software portability. Versions of NT family operating systems have been released for a variety of processor architectures, initially Intel IA-32, MIPS R3000/R4000 and Alpha, with PowerPC, Itanium and AMD64 supported in later releases.

Process Management: The Windows NT startup process is the process by which Microsoft's Windows NT, Windows 2000, Windows XP and Windows Server 2003 operating systems initialize.

Memory Management: On x86-based PCs, extended memory is only available with an Intel 80286 processor or higher. Only these chips can address more than 1 MB of RAM.

Key Points:

1. Windows XP

Windows XP is a line of operating systems produced by Microsoft for use on personal computers, including home and business desktops, notebook computers, and media centers.

The name "XP" is short for "experience". Windows XP is the successor to both Windows 2000 Professional and Windows Me, and is the first consumer-oriented operating system produced by Microsoft to be built on the Windows NT kernel and architecture. Windows XP was first released on 25 October 2001, and over 400 million copies were in use in January 2006, according to an estimate in that month by an IDC analyst. It is succeeded by Windows Vista, which was released to volume license customers on 8 November 2006 and worldwide to the general public on 30 January 2007. Direct OEM and retail sales of Windows XP ceased on 30 June 2008, although it is still possible to obtain Windows XP from System Builders (smaller OEMs who sell assembled computers) until 31 July 2009 or by purchasing Windows Vista Ultimate or Business and then downgrading to Windows XP. The most common editions of the operating system are Windows XP Home Edition, which is targeted at home users, and Windows XP Professional, which offers additional features such as support for Windows Server domains and two physical processors, and is targeted at power users, business and enterprise clients. Windows XP Media Center Edition has additional multimedia features enhancing the ability to record and watch TV shows, view DVD movies, and listen to music. Windows XP Tablet PC Edition is designed to run ink-aware applications built using the Tablet PC platform. Two separate 64-bit versions of Windows XP were also released, Windows XP 64-bit Edition for IA-64 (Itanium) processors and Windows XP Professional x64 Edition for x86-64. There is also Windows XP Embedded, a componentized version of the Windows XP Professional, and editions for specific markets such as Windows XP Starter Edition. Windows XP is known for its improved stability and efficiency over the 9x versions of Microsoft Windows. It presents a significantly redesigned graphical user interface, a change Microsoft promoted as more user-friendly than previous versions of Windows. A new software management facility called Side-by-Side Assembly was introduced to avoid the "DLL hell" that plagued older consumer-oriented 9x versions of Windows. It is also the first version of Windows to use product activation to combat software piracy, a restriction that did not sit well with some users and privacy advocates. Windows XP has also been criticized by some users for security vulnerabilities, tight integration of applications such as Internet Explorer 6 and Windows Media Player, and for aspects of its default user interface. Later versions with Service Pack 2, and Internet Explorer 7 addressed some of these concerns. During development, the project was codenamed "Whistler", after Whistler, British Columbia, as many Microsoft employees skied at the Whistler-Blackcomb ski resort. As of the end of November 2008, Windows XP is the most widely used operating system in the world with a 66.31% market share, having peaked at 85% in December 2006.

2. Windows XP Architecture

A main design goal of NT was hardware and software portability. Versions of NT family operating systems have been released for a variety of processor architectures, initially Intel IA-32, MIPS R3000/R4000 and Alpha, with PowerPC, Itanium and AMD64 supported in later releases. The idea was to have a common code base with a custom Hardware Abstraction Layer (HAL) for each platform. However, support for MIPS, Alpha and PowerPC was later dropped. Broad software compatibility was achieved with support for several API "personalities", including Win32, POSIX and OS/2 APIs. Partial MS-DOS compatibility was achieved via an integrated DOS Virtual Machine. NT supported per-object (file, function, and role) access control lists allowing a rich set of security permissions to be applied to systems and services. NT supported Windows network protocols, inheriting the previous OS/2 LAN Manager networking, as well as TCP/IP networking (for which Microsoft would implement a TCP/IP stack derived from the BSD UNIX stack). Windows NT 3.1 was the first version of Windows to utilize 32-bit "flat" virtual memory addressing on 32-bit processors. Its companion product, Windows 3.1, used segmented addressing and switches from 16-bit to 32-bit addressing in pages. Windows NT 3.1 featured a core kernel providing a system API, running in supervisor mode, and a set of user-space environments with their own APIs which included the new Win32 environment, an OS/2 1.3 text-mode environment and a POSIX environment. The full preemptive multitasking kernel could interrupt running tasks to schedule other tasks, without relying on user programs to voluntarily give up control of the CPU, as in Windows 3.1 Windows applications (although MS-DOS applications were preemptively multitasked in Windows starting with Windows 1.0).

Notably, in Windows NT 3.x, several I/O driver subsystems, such as video and printing, were user-mode subsystems. In Windows NT 4, the video, server and printer spooler subsystems were integrated into the kernel. Windows NT's first GUI was strongly influenced by (and programmatically compatible with) that from Windows 3.1; Windows NT 4's interface was redesigned to match that of the brand new Windows 95, moving from the Program Manager to the Start Menu/Taskbar design. NTFS, a journaled, secure file system, was created for NT. NT also allows for other installable file systems, and with versions 3.1 and 3.51, NT could also be installed on DOS's FAT or OS/2's HPFS file systems. Later versions could be installed on a FAT32 partition gaining speed at the expense of security, but this option is no longer present in Windows Vista. Microsoft decided to create a portable operating system, compatible with OS/2 and POSIX support and with multiprocessing in October 1988. When

development started in November 1989, Windows NT was to be known as OS/2 3.0, the third version of the operating system developed jointly by Microsoft and IBM. In addition to working on three versions of OS/2, Microsoft continued parallel development of the DOS-based and less resource-demanding Windows environment. When Windows 3.0 was released in May 1990, it was eventually so successful that Microsoft decided to change the primary application programming interface for the still unreleased NT OS/2 (as it was then known) from an extended OS/2 API to an extended Windows API.

This decision caused tension between Microsoft and IBM and the collaboration ultimately fell apart. IBM continued OS/2 development alone while Microsoft continued work on the newly renamed Windows NT. Though neither operating system would immediately be as popular as Microsoft's MS-DOS or Windows products, Windows NT would eventually be far more successful than OS/2. Microsoft hired a group of developers from Digital Equipment Corporation led by Dave Cutler to build Windows NT, and many elements of the design reflect earlier DEC experience with Cutler's VMS and RSX-11. The operating system was designed to run on multiple instruction set architectures and multiple hardware platforms within each architecture. The platform dependencies are largely hidden from the rest of the system by a kernel mode module called the HAL (Hardware Abstraction Layer). Windows NT's kernel mode code further distinguishes between the "kernel", whose primary purpose is to implement processor and architecture dependent functions, and the "executive". This was designed as a modified microkernel, as the Windows NT kernel does not meet all of the criteria of a pure microkernel. Both the kernel and the executive are linked together into the single loaded module ntoskrnl.exe; from outside this module there is little distinction between the kernel and the executive. Routines from each are directly accessible, as for example from kernel-mode device drivers. API sets in the Windows NT family are implemented as subsystems atop the publicly undocumented "native" API; it was this that allowed the late adoption of the Windows API (into the Win32 subsystem). Windows NT was one of the earliest operating systems to use Unicode internally.

3. Process Management.

The Windows NT startup process is the process by which Microsoft's Windows NT, Windows 2000, Windows XP and Windows Server 2003 operating systems initialize.
In Windows Vista, this process has changed slightly (see Windows Vista startup process). An NTLDR file, located in the root folder of the boot disk, is composed of two parts. The first is

the StartUp module and immediately followed by the OS loader (osloader.exe), both stored within that file. When NTLDR is loaded into memory and control is first passed to StartUp module, the CPU is operating in real mode. StartUp module's main task is to switch the processor into protected mode, which facilitates 32-bit memory access, thus allowing it to create the initial Interrupt descriptor table, Global Descriptor Table, page tables and enable paging. This provides the basic operating environment on which the operating system will build. StartUp module then loads and launches OS loader. NTLDR's OS loader includes basic functionality to access IDE-based disks formatted for NTFS or FAT file systems, or CDFS (ISO 9660), ETFS or UDFS in newer operating system versions. Disks are accessed through the system BIOS, through native ARC routines on ARC systems, or via network using TFTP protocol. All BIOS calls are done through virtual 8086 mode beyond this point, as the BIOS cannot be directly accessed within protected mode. If the boot disk is a SCSI disk and the SCSI controller is not using real-mode INT 0x13, an additional file, Ntbootdd.sys is loaded to handle disk access in place of the default routines. This is a copy of the same SCSI miniport driver that is used when Windows is running. The boot loader then reads the contents of boot.ini to locate information on the system volume. If the boot.ini file is missing, the boot loader will attempt to locate information from the standard installation directory. For Windows NT machines, it will attempt to boot from C:\WINNT. For Windows XP and 2003 machines, it will boot from C:\WINDOWS. At this point, the screen is cleared, and in the Windows 2000 or later versions of NTLDR and IA64ldr which support system hibernation, the root directory default volume as defined in boot.ini is searched for a hibernation file, hiberfil.sys. If this file is found and an active memory set is found in it, the contents of the file (which will match the amount of physical memory in the machine) are loaded into memory, and control is transferred into the Windows kernel at a point from which hibernation can be resumed. The file is then immediately marked as non-active, so that a crash or other malfunction cannot cause this (now-outdated) memory state to be re-loaded. If a state resume fails, the next time NTLDR runs it will ask the user whether to try resuming again or to discard the file and proceed with normal booting.

4. Memory Management

On x86-based PCs, extended memory is only available with an Intel 80286 processor or higher. Only these chips can address more than 1 MB of RAM. The earlier 8086/8088 processors can make use of more than 1 MB of RAM, if one employs special hardware to make selectable parts of it appear at addresses below 1 MB (paging). On a 286 or better PC equipped with more than 640 KB of RAM, the additional memory would generally be re-

mapped above the 1 MB boundary, since the IBM PC architecture mandates a 384 KB "hole" in memory between the 640 KB and 1 MB boundaries. This way all of the additional memory would be available to programs running in protected mode. Even without such remapping, machines with more than 1 MB of RAM would have access to memory above 1 MB. Extended memory is available in real mode only through EMS, UMB, XMS, or HMA; only applications executing in protected mode can use extended memory directly. In this case, the extended memory is provided by a supervising protected-mode operating system such as Microsoft Windows. The processor makes this memory available through the Global Descriptor Table and one or more Local Descriptor Tables (LDTs). The memory is "protected" in the sense that memory segments assigned a local descriptor cannot be accessed by another program because that program uses a different LDT, and memory segments assigned a global descriptor can have their access rights restricted, causing a hardware trap (typically a General Protection Fault) on violation. This prevents programs running in protected mode from interfering with each other's memory. A protected-mode operating system such as Microsoft Windows can also run real-mode programs and provide expanded memory to them. The DOS Protected Mode Interface is Microsoft's prescribed method for an MS-DOS program to access extended memory under a multitasking environment. Extended Memory Specification or XMS is the specification describing the use of IBM PC extended memory in real mode for storing data (but not for running executable code in it). Memory is made available by extended memory manager (XMM) software such as HIMEM.SYS. The XMM functions are accessible through interrupt 2Fh. XMS should not be confused with the somewhat similar EMS (expanded memory specification).

5. Disk Management

The Logical Disk Manager is an implementation of a logical volume manager for Microsoft Windows NT, developed by Microsoft and Veritas Software. It was introduced with the Windows 2000 operating system, and is supported in Windows XP, Windows Server 2003 and Windows Vista. Basic storage involves dividing a disk into primary and extended partitions. This is the way that all versions of Windows that were reliant on DOS handled storage took, and disks formatted in this manner are known as basic disks. Dynamic storage involves the use of a single partition that covers the entire disk, and the disk itself is divided into volumes or combined with other disks to form volumes that are greater in size than one disk itself. Volumes can use any supported file system. Basic disks can be upgraded to

dynamic disks, however when this is done the disk cannot easily be downgraded to a basic disk again. To perform a downgrade, data on the dynamic disk must first be backed up onto some other storage device. Second, the dynamic disk must be re-formatted as a basic disk (erasing all data). Finally, data from the backup must be copied back over to the newly re-formatted basic disk. Dynamic disks provide the capability for software implementations of RAID. The main disadvantage of dynamic disks in Microsoft Windows is that they can only be recognized under certain operating systems, such as Windows 2000 or later (excluding home versions such as Windows XP Home Edition, and Windows Vista Home Basic and Premium), or the Linux kernel starting with version 2.4.8. Dynamic disks under Windows are provided with the use of databases stored on disk(s). The volumes are referred to as dynamic volumes. It is possible to have 2000 dynamic volumes per dynamic disk, but the maximum recommended by Microsoft is 32.

6. File Management

File Manager is a file manager program bundled with releases of Microsoft Windows between 1990 and 1999. It was intended to replace using MS-DOS to manage the user's files. File Manager was retired in favor of Windows Explorer with the release of Windows 95 and Windows NT 4.0. The program's interface showed a list of directories (later called folders) on the left side, and a list of the current directory's contents on the right side. File Manager allowed a user to create, rename, move, print, copy, search for, and delete files and directories, as well as to set permissions such as read-only or hidden, and to associate file types with programs. Also available were tools to label and format disks and to connect and disconnect from a network drive. On Windows NT systems it was also possible to set ACLs on files and folders on NTFS partitions through the shell32 security configuration dialog (also used by Explorer and other Windows file managers). The Windows NT version of File Manager allows users to change file and directory permissions. This is not possible with Windows Explorer on Windows XP Home Edition as users are restricted to Simple File Sharing (unless running in Safe Mode). From Windows 95 and Windows NT 4.0 onward, File Manager was superseded by Windows Explorer. However, the WINFILE.EXE program file was still included with Windows 95, Windows NT 4.0, Windows 98, and Windows Me. Ian Ellison-Taylor was the shell developer on the Windows 3.1 team responsible for File Manager and Print Manager.

7. 16-bit

The original version of File Manager was a 16 bit program that supported the 8.3 file names that were in use at the time. It did not support the extended file names that became available in Windows 95 - including long file names and file names containing spaces. Instead, it would show these file names with a 'tilde' character "~" in the file The 16-bit version had a Y2K issue due to lexicographic correlation between dates and the ASCII character set; colons and semicolons replaced what should have been '2000'. File Manager was completely rewritten as a 32-bit application for Windows NT. This new version correctly handled long file names as well as NTFS file systems. It was included with Windows NT 3.1, 3.5, 3.51, and 4.0. It is possible to run File Manager on Windows 2000 and Windows XP by extracting a copy of the WINFILE.EXE program file from a Windows NT 4.0 Installation, Windows NT 4.0 CD-ROM, or by downloading and expanding the files from Windows NT4.0 Service Pack 6. File Manager cannot run natively under Windows Vista because it does not include COMMCTRL.DLL. However, it is possible to create a modified version that will run on Vistausing instructions available here

8. The Registry

The Windows Registry is a database which stores settings and options for Microsoft Windows operating systems. It contains information and settings for all the hardware, operating system software, most non-operating system software, and per-user settings. The registry also provides a window into the operation of the kernel, exposing runtime information such as performance counters and currently active hardware. When first introduced with Windows 3.1, the Windows registry's purpose was to tidy up the profusion of per-program INI files that had previously been used to store configuration settings for Windows programs. These files tended to be scattered all over the system, which made them difficult to track.

9. Keys and values

The registry contains two basic elements: keys and values.

Registry Keys are similar to folders - in addition to values; each key can contain subkeys, which may contain further subkeys, and so on. Keys are referenced with syntax similar to Windows' path names, using backslashes to indicate levels of hierarchy. E.g. HKEY_LOCAL_MACHINE\Software\Microsoft\Windows refers to the subkey "Windows" of the subkey "Microsoft" of the subkey "Software" of the HKEY_LOCAL_MACHINE key.

RegistryValues are name/data pairs stored within keys. Values are referenced separately from keys. Value names can contain backslashes but doing so makes them difficult to distinguish from their key paths. The Windows API functions that query and manipulate registry values take value names separately from the key path and/or handle that identify the parent key. The terminology is somewhat misleading, as the values are similar to an associative array, where standard terminology would refer to the name part of the value as a "key". The terms are a holdout from the 16-bit registry in Windows 3, in which keys could not contain arbitrary name/data pairs, but rather contained only one unnamed value (which had to be a string). In this sense, the entire registry was like an associative array where the keys (in both the registry sense and dictionary sense) formed a hierarchy, and the values were all strings. When the 32-bit registry was created, so was the additional capability of creating multiple named values per key, and the meanings of the names were somewhat distorted.

10. Hives

The Registry is split into a number of logical sections, or "hives" (the reason the word hive was used is an in-joke) Hives are generally named by their Windows API definitions, which all begin "HKEY". They are abbreviated to a three- or four-letter short name starting with "HK" (e.g. HKCU and HKLM). The HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER nodes have a similar structure to each other; applications typically look up their settings by first checking for them in "HKEY_CURRENT_USER\Software\Vendor's name\Application's name\Version\Setting name", and if the setting is not found look instead in the same location under the HKEY_LOCAL_MACHINE key. When writing settings back, the reverse approach is used HKEY_LOCAL_MACHINE is written first, but if that cannot be written to (which is usually the case if the logged-in user is not an administrator), the setting is stored in HKEY_CURRENT_USER instead. HKEY_CLASSES_ROOT (HKCR) Abbreviated HKCR, HKEY_CLASSES_ROOT stores information about registered applications, such as file associations and OLE Object Class IDs tying them to the applications used to handle these items. On Windows 2000 and above, HKCR is a compilation of HKCU\Software\Classes and HKLM\Software\Classes.

If a given value exists in both of the subkeys above, the one in HKCU\Software\Classes is used. HKEY_CURRENT_USER (HKCU) Abbreviated HKCU, HKEY_CURRENT_USER stores settings that are specific to the currently logged-in user. The HKCU key is a link to the subkey of HKEY_USERS that corresponds to the user; the same information is reflected in

both locations. On Windows-NT based systems, each user's settings are stored in their own files called NTUSER.DAT and USRCLASS.DAT inside their own Documents and Settings subfolder (or their own Users subfolder in Windows Vista). Settings in this hive follow users with a roaming profile from machine to machine. HKEY_LOCAL_MACHINE (HKLM) Abbreviated HKLM, HKEY_LOCAL_MACHINE stores settings that are general to all users on the computer. On NT-based versions of Windows, HKLM contains four subkeys, SAM, SECURITY, SOFTWARE and SYSTEM, which are found within their respective files located in the %SystemRoot%\System32\config folder. A fifth subkey, HARDWARE, is volatile and is created dynamically, and as such is not stored in a file. Information about system hardware drivers and services are located under the SYSTEM subkey, while the SOFTWARE subkey contains software and Windows settings. HKEY_CURRENT_CONFIG Abbreviated HKCC, HKEY_CURRENT_CONFIG contains information gathered at runtime; information stored in this key is not permanently stored on disk, but rather regenerated at the boot time. HKEY_PERFORMANCE_DATA This key provides runtime information into performance data provided by either the NT kernel itself or other programs that provide performance data. This key is not displayed in the Registry Editor, but it is visible through the registry functions in the Windows API. HKEY_DYN_DATA This key is used only on Windows 95, Windows 98 and Windows Me. It contains information about hardware devices, including Plug-and-Play and network performance statistics. The information in this hive is also not stored on the hard drive. The Plug and Play information is gathered and configured at startup and is stored in memory.

11. Symbolic Links

In Windows NT based systems Symbolic Links between registry keys are supported through REG_LINK value type. Registry links work similarly to file shortcuts or filesystem Symbolic links. As such they can span across different hives, however only those visible in Native API namespace, that is \Registry\Machine and \Registry\User. Other hives like HKEY_DYN_DATA are only virtual objects in Win32 API and thus not linkable. Links are used in Windows rather scarcely, only by CurrentControlSetand Hardware Profiles\Current.

12. 64-bit operating systems

Windows NT included support for several different platforms before the x86-based personal computer became dominant in the professional world. Versions of NT from 3.1 to 4.0 variously supported PowerPC, DEC Alpha and MIPS R4000, some of which were 64-bit processors, although the operating system treated them as 32-bit processors. With the introduction of the Intel Itanium architecture, which is referred to as IA-64, Microsoft

released new versions of Windows to support it. Itanium versions of Windows XP and Windows Server 2003 were released at the same time as their mainstream x86 (32-bit) counterparts. On April 25, 2005, Microsoft released Windows XP Professional x64 Edition and x64 versions of Windows Server 2003 to support the AMD64/Intel64 (or x64 in Microsoft terminology) architecture. Microsoft dropped support for the Itanium version of Windows XP in 2005. Windows Vista is the first end-user version of Windows that Microsoft has released simultaneously in 32-bit and x64 editions. Windows Vista does not support the Itanium architecture. The modern 64-bit Windows family comprises AMD64/Intel64 versions of Windows Vista, and Windows Server 2003 and Windows Server 2008, in both Itanium and x64 editions.

Topic : Unixand Linux Internals

Topic Objective:

At the end of this topic student will be able to understand:

- Understand the UNIXSystem
- Understand the Images and Processes
- Understand the Time-slicing and Interrupts
- Understand the Encoding time as a number
- Understand the Memory Management
- Understand the File System
- Understand the Linux
- Understand the Design

Definition/Overview:

Linux: Linux is the name usually given to any Unix-like computer operating system that uses the Linux kernel. Linux is one of the most prominent examples of free software and open source development: typically all underlying source code can be freely modified, used, and redistributed by anyone.

UNIX:UNIX (officially trademarked as UNIX, sometimes also written as UNIX or Unix with small caps) is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs including Ken Thompson, Dennis Ritchie and Douglas McIlroy. Today's UNIX systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organizations.

Key Points:

1. The UNIXSystem

As of 2007, the owner of the trademark is The Open Group, an industry standards consortium. Only systems fully compliant with and certified to the Single UNIX Specification are qualified to use the trademark; others are called "UNIX system-like" or "Unix-like". During the late 1970s and early 1980s, the influence of UNIX in academic circles led to large-scale adoption of UNIX (particularly of the BSD variant, originating from the University of California, Berkeley) by commercial startups, the most notable of which are Solaris, HP-UX, and AIX. Today, in addition to certified UNIX systems, Unix-like operating systems such as Linux and BSD are commonly encountered. Sometimes, "traditional Unix" may be used to describe a UNIX or an operating system that has the characteristics of either Version 7 UNIX or UNIX System V.

It was written in high level language as opposed to assembly language (which had been thought necessary for systems implementation on early computers). Although this followed the lead of Multics and Burroughs, it was UNIX that popularized the idea. UNIX had a drastically simplified file model compared to many contemporary operating systems, treating all kinds of files as simple byte arrays. The file system hierarchy contained machine services and devices (such as printers, terminals, or disk drives), providing a uniform interface, but at the expense of occasionally requiring additional mechanisms such as ioctl and mode flags to access features of the hardware that did not fit the simple "stream of bytes" model. The Plan 9 operating system pushed this model even further and eliminated the need for additional mechanisms. UNIX also popularized the hierarchical file system with arbitrarily nested subdirectories, originally introduced by Multics. Other common operating systems of the era had ways to divide a storage device into multiple directories or sections, but they had a fixed number of levels, often only one level. Several major proprietary operating systems eventually added recursive subdirectory capabilities also patterned after Multics. DEC's RSX-11M's "group, user" hierarchy evolved into VMS directories, CP/M's volumes evolved into

MS-DOS 2.0+ subdirectories, and HP's MPE group.account hierarchy and IBM's SSP and OS/400 library systems were folded into broader POSIX file systems. Making the command interpreter an ordinary user-level program, with additional commands provided as separate programs, was another Multics innovation popularized by UNIX. The UNIX shell used the same language for interactive commands as for scripting (shell scripts there was no separate job control language like IBM's JCL). Since the shell and OS commands were "just another program", the user could choose (or even write) his own shell. New commands could be added without changing the shell itself. UNIXs innovative command-line syntax for creating chains of producer-consumer processes (pipelines) made a powerful programming paradigm (coroutines) widely available. Many later command-line interpreters have been inspired by the UNIX shell.

A fundamental simplifying assumption of UNIX was its focus on ASCII text for nearly all file formats. There were no "binary" editors in the original version of UNIX the entire system was configured using textual shell command scripts. The common denominator in the I/O system was the byte unlike "record-based" file systems. The focus on text for representing nearly everything made UNIX pipes especially useful, and encouraged the development of simple, general tools that could be easily combined to perform more complicated ad hoc tasks. The focus on text and bytes made the system far more scalable and portable than other systems. Over time, text-based applications have also proven popular in application areas, such as printing languages (PostScript, ODF), and at the application layer of the Internet protocols, e.g., Telnet, FTP, SSH, SMTP, HTTP, SOAP and SIP. UNIX popularized syntax for regular expressions that found widespread use. The UNIX programming interface became the basis for a widely implemented operating system interface standard (POSIX, see above). The C programming language soon spread beyond UNIX, and is now ubiquitous in systems and applications programming. Early UNIX developers were important in bringing the concepts of modularity and reusability into software engineering practice, spawning a "software tools" movement. UNIX provided the TCP/IP networking protocol on relatively inexpensive computers, which contributed to the Internet explosion of worldwide real-time connectivity, and which formed the basis for implementations on many other platforms. This also exposed numerous security holes in the networking implementations. The UNIX policy of extensive on-line documentation and (for many years) ready access to all system source code raised programmer expectations, and contributed to the 1983 launch of the free software movement.

Over time, the leading developers of UNIX (and programs that ran on it) established a set of cultural norms for developing software, norms which became as important and influential as the technology of UNIX itself; this has been termed the UNIX philosophy.

2. Images and Processes

In UNIX and other computer multitasking operating systems, a daemon (pronounced /ˈdiːmən/ or /ˈdeɪmən/) is a computer program that runs in the background, rather than under the direct control of a user; they are usually initiated as background processes. Typically daemons have names that end with the letter "d": for example, syslogd, and the daemon that handles the system log, or sshd, which handles incoming SSH connections. In a UNIX environment, the parent process of a daemon is often (but not always) the init process (PID=1). Processes usually become daemons by forking a child process and then having their parent process immediately exit, thus causing init to adopt the child process. This is a somewhat simplified view of the process as other operations are generally performed, such as disassociating the daemon process from any controlling tty. Convenience routines such as daemon(3) exist in some UNIX systems for that purpose. Systems often start (or "launch") daemons at boot time: they often serve the function of responding to network requests, hardware activity, or other programs by performing some task. Daemons can also configure hardware (like devfsd on some Linux systems), run scheduled tasks (like cron), and perform a variety of other tasks. The term was coined by the programmers of MIT's Project MAC. They took the name from Maxwell's demon, an imaginary being from a famous thought experiment that constantly works in the background, sorting molecules. UNIX systems inherited this terminology. Daemons are also characters in Greek mythology, some of whom handled tasks that the gods could not be bothered with, much as computer daemons often handle tasks in the background that the user cannot be bothered with. BSD and some of its derivatives have adopted a daemon as its mascot, although this mascot is actually a cute variation of the demons which appear in Christian artwork.

3. Time-slicing and Interrupts.

UNIX time, or POSIX time, is a system for describing points in time, defined as the number of seconds elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds. It is widely used not only on Unix-like operating systems but also in many other computing systems. It is neither a linear representation of time nor a true

representation of UTC (though it is frequently mistaken for both) as the times it represents are UTC but it has no way of representing UTC leap seconds (e.g. 1998-12-31 23:59:60).

4. Encoding time as a number

Modern UNIX time is based strictly on UTC. UTC counts time using SI seconds, and breaks up the span of time into days. UTC days are mostly 86400 s long, but due to "leap seconds" are occasionally 86401 s and could be 86399 s long (though the latter option has never been used as of January 2009); this keeps the days synchronized with the rotation of the Earth (or Universal Time). As is standard with UTC, this article labels days using the Gregorian calendar, and counts times within each day in hours, minutes, and seconds. Some of the examples also show TAI, another time scheme, which uses the same seconds and is displayed in the same format as UTC, but has every day exactly 86400 s long, making no attempt to stay synchronized with the Earth's rotation. The UNIX epoch is the time 00:00:00 UTC on January 1, 1970. There is a problem with this definition, in that UTC did not exist in its current form until 1972; this issue is discussed below. For brevity, the remainder of this section uses ISO 8601 date format, in which the UNIXepoch is 1970-01-01T00:00:00Z. The UNIX time number is zero at the UNIX epoch, and increases by exactly 86 400 per day since the epoch. Thus 2004-09-16T00:00:00Z, 12 677 days after the epoch, is represented by the UNIX time number 12 677 86 400 = 1 095 292 800. This can be extended backwards from the epoch too, using negative numbers; thus 1957-10-04T00:00:00Z, 4472 days before the epoch, is represented by the UNIX time number -4472 86 400 = -386 380 800. Within each day, the UNIX time number is as calculated in the preceding paragraph at midnight UTC (00:00:00Z), and increases by exactly 1 per second since midnight. Thus 2004-09-16T17:55:43.54Z, 64 543.54 s since midnight on the day in the example above, is represented by the UNIX time number 1 095 292 800 + 64 543.54 = 1 095 357 343.54. On dates before the epoch the number still increases, thus becoming less negative, as time moves forward. The above scheme means that on a normal UTC day, of duration 86 400 s, the UNIX time number changes in a continuous manner across midnight. For example, at the end of the day used in the examples above, the time representations progress like this:

UNIXTIME ACROSS MIDNIGHT ON A NORMAL UTC DAY		
TAI	**UTC**	**UNIXtime**
2004-09-17T00:00:30.75	2004-09-16T23:59:58.75	1 095 379 198.75
2004-09-17T00:00:31.00	2004-09-16T23:59:59.00	1 095 379 199.00

2004-09-17T00:00:31.25	2004-09-16T23:59:59.25	1 095 379 199.25
2004-09-17T00:00:31.50	2004-09-16T23:59:59.50	1 095 379 199.50
2004-09-17T00:00:31.75	**2004-09-16T23:59:59.75**	**1 095 379 199.75**
2004-09-17T00:00:32.00	**2004-09-17T00:00:00.00**	**1 095 379 200.00**
2004-09-17T00:00:32.25	2004-09-17T00:00:00.25	1 095 379 200.25
2004-09-17T00:00:32.50	2004-09-17T00:00:00.50	1 095 379 200.50
2004-09-17T00:00:32.75	2004-09-17T00:00:00.75	1 095 379 200.75
2004-09-17T00:00:33.00	2004-09-17T00:00:01.00	1 095 379 201.00
2004-09-17T00:00:33.25	2004-09-17T00:00:01.25	1 095 379 201.25

[Table 1]

When a leap second occurs, so that the UTC day is not exactly 86 400 s long, a discontinuity occurs in the UNIX time number. The UNIX time number increases by exactly 86 400 each day, regardless of how long the day is. When a leap second is deleted (which has never occurred as of 2008), the UNIX time number jumps up by 1 at the instant where the leap second was deleted from, which is the end of the day. When a leap second is inserted (which has occurred on average once every year and a half), the UNIX time number increases continuously during the leap second, during which time it is more than 86 400 s since the start of the current day, and then jumps down by 1 at the end of the leap second, which is the start of the next day. For example, this is what happened on strictly conforming POSIX.1 systems at the end of 1998:

UNIXTIME ACROSS MIDNIGHT WHEN A UTC LEAP SECOND IS INSERTED		
TAI	**UTC**	**UNIXtime**
1999-01-01T00:00:29.75	1998-12-31T23:59:58.75	915 148 798.75
1999-01-01T00:00:30.00	1998-12-31T23:59:59.00	915 148 799.00
1999-01-01T00:00:30.25	1998-12-31T23:59:59.25	915 148 799.25
1999-01-01T00:00:30.50	1998-12-31T23:59:59.50	915 148 799.50
1999-01-01T00:00:30.75	**1998-12-31T23:59:59.75**	**915 148 799.75**
1999-01-01T00:00:31.00	**1998-12-31T23:59:60.00**	**915 148 800.00**
1999-01-01T00:00:31.25	1998-12-31T23:59:60.25	915 148 800.25
1999-01-01T00:00:31.50	1998-12-31T23:59:60.50	915 148 800.50
1999-01-01T00:00:31.75	**1998-12-31T23:59:60.75**	**915 148 800.75**
1999-01-01T00:00:32.00	**1999-01-01T00:00:00.00**	**915 148 800.00**

1999-01-01T00:00:32.25	1999-01-01T00:00:00.25	915 148 800.25
1999-01-01T00:00:32.50	1999-01-01T00:00:00.50	915 148 800.50
1999-01-01T00:00:32.75	1999-01-01T00:00:00.75	915 148 800.75
1999-01-01T00:00:33.00	1999-01-01T00:00:01.00	915 148 801.00
1999-01-01T00:00:33.25	1999-01-01T00:00:01.25	915 148 801.25

[Table 2]

Observe that when a positive leap second occurs (i.e., when a leap second is inserted) the UNIX time numbers repeat themselves. The UNIX time number 915 148 800.50 is ambiguous: it can refer either to the instant in the middle of the leap second, or to the instant one second later, half a second after midnight UTC. In the theoretical case when a negative leap second occurs (i.e., when a leap second is deleted) no ambiguity is caused, but instead there is a range of UNIX time numbers that do not refer to any point in time at all. A UNIX clock is often implemented with a different type of positive leap second handling associated with the Network Time Protocol (NTP). This yields a system that does not conform to the POSIX standard. See the section below concerning NTP for details. When dealing with periods that do not encompass a UTC leap second, the difference between two UNIX time numbers is equal to the duration in seconds of the period between the corresponding points in time. This is a common computational technique. However, where leap seconds occur, such calculations give the wrong answer. In applications where this level of accuracy is required, it is necessary to consult a table of leap seconds when dealing with UNIX times, and it is often preferable to use a different time encoding that does not suffer this problem. A UNIX time number is easily converted back into UTC by taking the quotient and modulus of the UNIX time number, modulo 86400. The quotient is the number of days since the epoch, and the modulus is the number of seconds since midnight UTC on that day. (It is important to ensure that the right type of modulus is being calculated when dealing with times before the epoch.) If given a UNIX time number that is ambiguous due to a positive leap second, this algorithm interprets it as the time just after midnight. It never generate a time that is during a leap second. If given a UNIX time number that is invalid due to a negative leap second, it generates an equally invalid UTC time. If these conditions are significant, it is necessary to consult a table of leap seconds to detect them.

5. Memory Management

A UNIX domain socket or IPC socket (inter-process communication socket) is a data communications endpoint that is similar to an Internet socket, but does not use a network protocol for communication. It is used in POSIX operating systems for inter-process communication. The correct standard POSIX term is POSIX Local IPC Sockets. UNIX domain connections appear as byte streams, much like network connections, but all data remains within the local computer. UNIX domain sockets use the file system as address name space, i.e. they are referenced by processes as inodes in the file system. This allows two distinct processes to open the same socket in order to communicate. However, the actual communication (the data exchange) does not use the file system, but buffers in kernel memory. In addition to sending data, processes can send file descriptors across a UNIX domain socket connection using the sendmsg() and recvmsg() system calls.

6. The File System

The UNIX file system (UFS) is a file system used by many UNIX and Unix-like operating systems. It is also called the Berkeley Fast File System, the BSD Fast File System or FFS. It is a distant descendant of the original filesystem used by Version 7 Unix.

A UFS volume is composed of the following parts:

- a few blocks at the beginning of the partition reserved for boot blocks (which must be initialized separately from the filesystem)
- a superblock, containing a magic number identifying this as a UFS filesystem, and some other vital numbers describing this filesystem's geometry and statistics and behavioral tuning parameters
- a collection of cylinder groups. Each cylinder group has the following components:
- a backup copy of the superblock
- a cylinder group header, with statistics, free lists, etc, about this cylinder group, similar to those in the superblock
- a number of inodes, each containing file attributes
- a number of data blocks

Vendors of some commercial UNIX systems, such as SunOS/Solaris, System V Release 4, HP-UX, and Tru64 UNIX, have adopted UFS. Most of them adapted UFS to their own uses, adding proprietary extensions that may not be recognized by other vendors' versions of UNIX. Surprisingly, many have continued to use the original block size and data field widths as the original UFS, so some degree of (read) compatibility remains across platforms. Compatibility between implementations as a whole is spotty at best and should be researched before using it across multiple platforms where shared data is a primary intent. As of Solaris 7, Sun Microsystems included UFS Logging, which brought filesystem journaling to UFS. Solaris UFS also has extensions for large files and large disks and other features. In 4.4BSD and BSD UNIXsystems derived from it, such as FreeBSD, NetBSD, OpenBSD, and DragonFlyBSD, the implementation of UFS1 and UFS2 is split into two layers an upper layer that provides the directory structure and supports metadata (permissions, ownership, etc.) in the inode structure, and lower layers that provide data containers implemented as inodes. This was done to support both the traditional FFS and the LFS log-structured file system with common code for common functions. The upper layer is called "UFS", and the lower layers are called "FFS" and "LFS". In some of those systems, the term "FFS" is used for the combination of the FFS lower layer and the UFS upper layer, and the term "LFS" is used for the combination of the LFS lower layer and the UFS upper layer. Kirk McKusick extended the FreeBSD FFS and UFS layers to support a new variant, called UFS2, which adds 64-bit block pointers (allowing volumes to grow up to 8 zettabytes), variable-sized blocks (similar to extents), extended flag fields, additional 'birthtime' stamps, extended attribute support and POSIX1.e ACLs. UFS2 became the default UFS version starting with FreeBSD 5.0. FreeBSD also introduced soft updates and the ability to make file system snapshots for both UFS1 and UFS2. These have since been ported to NetBSD. OpenBSD has supported soft updates since version 2.9 and UFS2 since version 4.2. Since FreeBSD 7.0, UFS also fully supports filesystem journaling using gjournal mechanism. Linux includes a UFS implementation for binary compatibility at the read level with other Unixes, but since there is no standard implementation for the vendor extensions to UFS, Linux does not have full support for writing to UFS. The native Linux ext2 filesystem is inspired by UFS. (In fact, in some 4.4BSD-derived systems, the UFS layer can use an ext2 layer as a container layer, just as it can use FFS and LFS.) NeXTStep, which was BSD-derived, also used a version of UFS. In Apple's Mac OS X, it is available as an alternative to HFS+, their proprietary filesystem. However, as of Mac OS X v10.5, one cannot install Mac OS X "Leopard" on a UFS-

formatted volume. In addition, one cannot upgrade older versions of Mac OS X installed on UFS-formatted volumes to Leopard; upgrading requires reformatting the startup volume.

6. Linux.

Linux is a generic term referring to Unix-like computer operating systems based on the Linux kernel. Their development is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed by anyone under the terms of the GNU GPL and other free licenses. Linux distributions are predominantly known for their use in servers, although they are installed on a wide variety of computer hardware, ranging from embedded devices and mobile phones to supercomputers, and their popularity as a desktop/laptop operating system has been growing lately due to the rise of netbooks and the Ubuntu distribution of the operating system. The name "Linux" comes from the Linux kernel, originally written in 1991 by Linus Torvalds. The rest of the system, including utilities and libraries, usually comes from the GNU operating system announced in 1983 by Richard Stallman. The GNU contribution is the basis for the alternative name **GNU/Linux**.

7. Design

A Linux-based system is a modular Unix-like operating system. It derives much of its basic design from principles established in UNIX during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, and peripheral and file system access. Device drivers are integrated directly with the kernel.

Separate projects that interface with the kernel provide much of the system's higher-level functionality. The GNU userland is an important part of most Linux-based systems, providing the most common implementation of the C library, a popular shell, and many of the common UNIX tools which carry out many basic operating system tasks. The graphical user interface on most Linux systems is based on the X Window System.

8. User interface

A Linux-based system can be controlled by one or more of a text-based command line interface (CLI), graphical user interface (GUI) (usually the default for desktop), or through controls on the device itself (common on embedded machines). On desktop machines, KDE, GNOME and Xfce are the most popular user interfaces, though a variety of other user interfaces exist. Most popular user interfaces run on top of the X Window System (X), which

provides network transparency, enabling a graphical application running on one machine to be displayed and controlled from another. Other GUIs include X window managers such as FVWM, Enlightenment and Window Maker. The window manager provides a means to control the placement and appearance of individual application windows, and interacts with the X window system. A Linux system typically provides a CLI of some sort through a shell, which is the traditional way of interacting with a UNIXsystem. A Linux distribution specialized for servers may use the CLI as its only interface. A headless system run without even a monitor can be controlled by the command line via a remote-control protocol such as SSH or telnet. Most low-level Linux components, including the GNU Userland, use the CLI exclusively. The CLI is particularly suited for automation of repetitive or delayed tasks, and provides very simple inter-process communication. A graphical terminal emulator program is often used to access the CLI from a Linux desktop.

In Section 4 of this course you will cover these topics:

- Macintosh Os X Internals
- Mvs Internals
- Data Communication And Networks
-

The Internet And The World Wide Web

Topic : Macintosh Os X Internals

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Mac OS X
Understand the Development outside of Apple
Understand the OS X Architecture.
Understand the NEXTSTEP
Understand the Rhapsody
Understand the Mac OS X
Understand the Darwin
Understand the Kernel
Understand the Memory Management

Definition/Overview:

OS X Architecture.: Mac OS X is the culmination of Apple Inc.'s decade-long search for an operating system to replace the original Mac OS.

NEXTSTEP: NEXTSTEP used a hybrid kernel that combined the Mach 2.5 kernel developed at Carnegie MellonUniversity with subsystems from 4.3BSD. NEXTSTEP also introduced a new windowing system based on Display PostScript that intended to achieve better WYSIWYG systems by using the same language to draw content on monitors that drew content on printers.
Rhapsody: On February 4, 1997, Apple acquired NeXT and began development of the Rhapsody operating system.

Key Points:

1. Mac OS X

Mac OS X is the newest of Apple Inc.'s Mac OS line of operating systems. Although it

is officially designated as simply "version 10" of the Mac OS, it has a history largely independent of the earlier Mac OS releases.

2. Development outside of Apple

After Apple removed Steve Jobs from management in 1985, he left the company and attempted with funding from Ross Perot and his own pockets to create the "next big thing": the result was NeXT. NeXT hardware was advanced for its time, being the first workstation to include a DSP and a high-capacity optical disc drive, but it had several quirks and design problems and was expensive compared to the rapidly commoditizing workstation market. The hardware was phased out in 1993. However, the company's object-oriented operating system NeXTSTEP had a more lasting legacy. NeXTSTEP was based on the Mach kernel and BSD, an implementation of UNIXdating back to the 1970s. Perhaps more remarkably, it featured an object-oriented programming framework based on the Objective-C language. This environment is known today in the Mac world as Cocoa. It also supported the innovative Enterprise Objects Framework database access layer and WebObjects application server development environment, among other notable features.

All but abandoning the idea of an operating system, NeXT managed to maintain a business selling WebObjects and consulting services, but was never a commercial success. NeXTSTEP underwent an evolution into OPENSTEP which separated the object layers from the operating system below, allowing it to run with less modification on other platforms. OPENSTEP was, for a short time, adopted by Sun Microsystems. However, by this point, a number of other companies notably Apple, IBM, Microsoft, and even Sun itself were claiming they would soon be releasing similar object-oriented operating systems and development tools of their own. (Some of these efforts, such as Taligent, did not fully come to fruition; others, like Java, gained widespread adoption.) Following an announcement on December 20, 1996, on February 4, 1997 Apple Computer acquired NeXT for $427 million, and used OPENSTEP as the basis for Mac OS X. Traces of the NeXT software heritage can still be seen in Mac OS X. For example, in the Cocoa development environment, the Objective-C library classes have "NS" prefixes, and the HISTORY section of the manual page for the defaultscommand in Mac OS X straightforwardly states that the

command "First appeared in NeXTStep."

3. OS X Architecture.

Mac OS X is the culmination of Apple Inc.'s decade-long search for an operating system to replace the original Mac OS. After the failures of their previous attempts; Pink which started as an Apple project but evolved into a joint venture with IBM called Taligent, and Copland that started in 1994 and was cancelled two years later, Apple began development of their most recent operating system (Mac OS X) with the acquisition of NeXT's NEXTSTEP.

4. NEXTSTEP

NEXTSTEP used a hybrid kernel that combined the Mach 2.5 kernel developed at Carnegie Mellon Universitywith subsystems from 4.3BSD. NEXTSTEP also introduced a new windowing system based on Display PostScript that intended to achieve better WYSIWYG systems by using the same language to draw content on monitors that drew content on printers. NeXT also included object-oriented programming tools based on the Objective-C language that they had acquired from Stepstone and a collection of Frameworks (or Kits) that were intended to speed software development. NEXTSTEP originally ran on Motorola's 68k processors, but was later ported to Intel's x86, Hewlett-Packard's PA-RISC and Sun Microsystems' SPARC processors. Later on, the developer tools and frameworks were released, as OpenStep, as a development platform that would run on other operating systems.

5. Rhapsody

On February 4, 1997, Apple acquired NeXT and began development of the Rhapsody operating system. Rhapsody built on NEXTSTEP, porting the core system to the PowerPC architecture and adding a redesigned user interface based on the Platinum user interface from Mac OS 8. An emulation layer called Blue Box allowed Mac OS applications to run within an actual instance of the Mac OS and an integrated Java platform. The Objective-C developer tools and Frameworks were referred to as the Yellow Box and also made available separately for Microsoft Windows. The Rhapsody project eventually bore the fruit of all Apple's efforts to develop a new generation Macintosh OS, which finally shipped in the form of Mac OS X Server.

6. Mac OS X

At the 1998 Worldwide Developers Conference (WWDC), Apple announced a move that was intended as a response to complaints from Macintosh software developers who were not happy with the two options (Yellow Box and Blue Box) available in Rhapsody. Mac OS X would add another developer API to the existing ones in Rhapsody. Key APIs from the Macintosh Toolbox would be implemented in Mac OS X to run directly on the BSD layers of the operating system instead of in the emulated Macintosh layer. This modified interface, called Carbon, would eliminate approximately 2000 troublesome API calls (of about 8000 total) and replace them with calls compatible with a modern OS.

At the same conference, Apple announced that the Mach side of the kernel had been updated with sources from version 3 of the Mach kernel and the BSD side of the kernel had been updated with sources from the FreeBSD, NetBSD and OpenBSD projects. They also announced a new driver model called I/O Kit, intended to replace the Driver Kit used in NEXTSTEP citing Driver Kit's lack of power management and hot-swap capabilities and its lack of automatic configuration capability. At the 1999 WWDC, Apple revealed Quartz, a new Portable Document Format (PDF) based windowing system for the operating system that wasn't encumbered with licensing fees to Adobe like the Display PostScript windowing system of NEXTSTEP. Apple also announced that the Yellow Box layer had been renamed Cocoa and began to move away from their commitment to providing the Yellow Box on Windows. At this WWDC, Apple also showed Mac OS X booting off of a HFS Plus formatted drive for the first time.

7.Darwin

Darwin is an open source POSIX-compliant computer operating system released by Apple Inc. in 2000. It is composed of code developed by Apple, as well as code derived from NEXTSTEP, FreeBSD, and other free software projects. Darwin forms the core set of components upon which Mac OS X and iPhone OS are based. It is compatible with the Single UNIXSpecification version 3 (SUSv3) and POSIX UNIXapplications and utilities.

8. Kernel

Darwin is built around XNU, a hybrid kernel that combines the Mach 3 microkernel, various elements of BSD (including the process model, network stack, and virtual file system),and an object-oriented device driver API called I/O Kit. Some of the benefits of this choice of kernel are the Mach-O binary format, which allows a single executable file (including the kernel itself) to support multiple CPU architectures, and the mature support for symmetric multiprocessing in Mach. The hybrid kernel design compromises between the flexibility of a microkernel and the performance of a monolithic kernel. Darwin currently includes support for both 32-bit and 64-bit variants of the PowerPC and Intel x86 processors used in the Mac and Apple TV as well as the 32-bit ARM processor used in the iPhone and iPod Touch. It supports the POSIX API by way of its BSD lineage and a large number of programs written for various other UNIX-like systems can be compiled on Darwinwith no changes to the source code. Darwin and Mac OS X both use I/O Kit for their drivers and therefore support the same hardware, file systems, and so forth. Apple's distribution of Darwin included proprietary (binary-only) drivers for their AirPort wireless cards. Darwin does not include many of the defining elements of Mac OS X, such as the Carbon and Cocoa APIs or the Quartz Compositor and Aqua user interface, and thus cannot run Mac applications. It does, however, support a number of lesser known features of Mac OS X, such as mDNSResponder, which is the multicast DNS responder and a core component of the Bonjour networking technology, and launched, an advanced service management framework.

9. Memory Management

Historically, the Mac OS used a form of memory management that has fallen out of favor in modern systems. Criticism of this approach was one of the key areas addressed by the change to Mac OS X. The original problem for the designers of the Macintosh was how to make optimum use of the 128 kB of RAM that the machine was equipped with. Since at that time the machine could only run one application program at a time, and there was no permanent secondary storage, the designers implemented a simple scheme which worked well with those particular constraints. However, that design choice did not scale well with the development of the machine,

creating various difficulties for both programmers and users.

10. Fragmentation

The chief worry of the original designers appears to have been fragmentationthat is, repeated allocation and deallocation of memory through pointers leads to many small isolated areas of memory which cannot be used because they are too small, even though the total free memory may be sufficient to satisfy a particular request for memory. To solve this, Apple designers used the concept of a relocatable handle, a reference to memory which allowed the actual data referred to be moved without invalidating the handle. Apple's scheme was simple - a handle was simply a pointer into a (non relocatable) table of further pointers, which in turn pointed to the data. If a memory request required compaction of memory, this was done and the table, called the master pointer block, was updated. The machine itself implemented two areas in the machine available for this scheme - the system heap (used for the OS), and the application heap. As long as only one application at a time was run, the system worked well. Since the entire application heap was dissolved when the application quit, fragmentation was minimized. However, in addressing the fragmentation problem, all other issues were overlooked. The system heap was not protected from errant applications, and this was frequently the cause of major system problems and crashes. In addition, the handle-based approach also opened up a source of programming errors, where pointers to data within such relocatable blocks could not be guaranteed to remain valid across calls that might cause memory to move. In reality this was almost every system API that existed. Thus the onus was on the programmer not to create such pointers, or at least manage them very carefully. Since many programmers were not generally familiar with this approach, early Mac programs suffered frequently from faults arising from this - faults that very often went undetected until long after shipment. Palm OS and 16-bit Windows use a similar scheme for memory management. However, the Palm and Windows versions make programmer error more difficult. For instance, in Mac OS to convert a handle to a pointer, a program just de-references the handle directly. However, if the handle is not locked, the pointer can become invalid quickly. Calls to lock and unlock handles are not balanced; ten calls to HLock are undone by a single call to HUnlock. In Palm OS and Windows, handles are opaque type and must be de-referenced with MemHandleLock on Palm OS or Global/LocalLock on Windows. When a Palm or Windows application is finished

with a handle, it calls MemHandleUnlock or Global/LocalUnlock. Palm OS and Windows keeps a lock count for blocks; after three calls to MemHandleLock, a block will only become unlocked after three calls to MemHandleUnlock.

11. Switcher

The situation worsened with the advent of Switcher, which was a way for the Mac to run multiple applications at once. This was a necessary step forward for users, who found the one-app-at-a-time approach very limiting. However, because Apple was now committed to its memory management model, as well as compatibility with existing applications, it was forced to adopt a scheme where each application was allocated its own heap from the available RAM. The amount of actual RAM allocated to each heap was set by a value coded into each application, set by the programmer. Invariably this value wasn't enough for particular kinds of work, so the value setting had to be exposed to the user to allow them to tweak the heap size to suit their own requirements. This exposure of a technical implementation detail was very much against the grain of the Mac user philosophy. Apart from exposing users to esoteric technicalities, it was inefficient, since an application would grab (unwillingly) its entire allotted RAM, even if it left most of it subsequently unused. Another application might be memory starved, but was unable to utilize the free memory "owned" by another application. Switcher became MultiFinder in System 4.2, which became the Process Manager in System 7, and by then the scheme was utterly entrenched. Apple made some attempts to work around the obvious limitations - temporary memory was one, where an application could "borrow" free RAM that lay outside of its heap for short periods, but this was unpopular with programmers so it largely failed to solve the problem. There was also an early virtual memory scheme, which attempted to solve the issue by making more memory available by paging unused portions to disk, but for most users with 68K Macintoshes, this did nothing but slow everything down without solving the memory problems themselves. By this time all machines had permanent hard disks and MMU chips, and so were equipped to adopt a far better approach. For some reason, Apple never made this a priority until Mac OS X, even though several schemes were suggested by outside developers that would retain compatibility while solving the overall memory management problem. Third party replacements for the Mac OS memory manager, such as Optimem, showed it could be done.

12. File Systems

Macintosh File System (**MFS**) is a volume format (or disk file system) created by Apple Computer for storing files on 400K floppy disks. MFS was introduced with the Macintosh 128K in January 1984. MFS was notable both for introducing resource forks to allow storage of structured data, and for storing metadata needed to support the graphical user interface of Mac OS. MFS allows file names to be up to 255 characters in length, although Finder does not allow users to create names longer than 63 characters (31 characters in later versions). MFS is called a flat file system because it does not support a hierarchy of directories. Folders existed as a concept on the original MFS-based Macintosh, but worked completely differently from the way they do on modern systems. They were visible in Finder windows, but not in the open and save dialog boxes. There was always one empty folder on the volume, and if it was altered in any way (such as by adding or renaming files), a new Empty Folder would appear, thus providing a way to create new folders. MFS stored all of the file and directory listing information in a single file. The Finder created the "illusion" of folders, by storing all files as a directory handle/file handle pair. To display the contents of a particular folder, MFS would scan the directory for all files in that handle. There was no need to find a separate file containing the directory listing. The Macintosh File System did not support volumes over 20 megabytes in size, or about 1,400 files. While this is small by today's standards, it seemed very expansive when all Apple Macintosh computers at the time had a 400 kilobyte floppy drive. Apple introduced Hierarchical File System as a replacement for MFS in September 1985. In Mac OS 7.6.1, Apple removed support for writing to MFS volumes, and in Mac OS 8 support for MFS volumes was removed altogether. Although Mac OS X has no built-in support for MFS, an example VFS plug-in from Apple called MFSLives provides read-only access to MFS volumes.

13. QuickTime

QuickTimeis a multimedia framework developed by Apple Inc., capable of handling various formats of digital video, media clips, sound, text, animation, music, and

interactive panoramic images. It is available for Mac OS (Mac OS 9, 8, 7, etc.), Mac OS X and Microsoft Windows operating systems.

QuickTime is integrated with Mac OS X, and it was an optional component at install for earlier versions of Mac OS. All Apple systems ship with QuickTime already installed. QuickTime for Windows systems is downloadable, either as a standalone installation or bundled with iTunes.
Software development kits (SDKs) for QuickTime are available to the public with a Apple Developer Connection (ADC) subscription.

14. QuickTime players

QuickTime is available for Mac OS X and Windows operating systems, and is distributed free of charge. Some other free player applications that rely on the QuickTime framework provide features not available in the basic QuickTime Player. For example:

iTunes can export audio in WAV, AIFF, MP3, AAC, and Apple Lossless.
In Mac OS X, a simple AppleScript can be used to play a movie in full-screen mode. However, since version 7.2 the QuickTime Player now also supports full-screen viewing in the non-pro version.

Any application can be written to access features provided by the QuickTime framework.

15. QuickTime Pro

The included QuickTime Player is limited to only the most basic playback operations unless the user purchases a QuickTime Pro license key, which Apple sells for US$29.95. Apple's "ProApplications" (e.g. Final Cut Studio, Logic Studio) come with a free QuickTime Pro license. Pro keys are specific to the major version of QuickTime for which they are purchased. The Pro key unlocks additional features of the QuickTime Player application on Mac OS X or Windows (although most of these can be accessed simply by using players, video editors or miscellaneous utilities from other sources). Use of the Pro key does not entail any additional downloads.
Features enabled by the Pro license include, but are not limited to:

Editing clips through the Cut, Copy and Paste functions, merging separate audio and

video tracks, and freely placing the video tracks on a virtual canvas with the options of cropping and rotation.
Saving and exporting (encoding) to any of the codecs supported by QuickTime.
QuickTime 7 includes presets for exporting video to a video-capable iPod, Apple TV, and the iPhone.
Saving existing QuickTime Movies from the web directly to a hard disk drive. This is often, but not always, either hidden or intentionally blocked in the standard mode. It should be noted that two options exist for saving movies from a web browser:
Save As Source - This option will save the embedded video in its original format. (i.e., not limited to *.mov files.)
Save As QuickTime Movie - This option will save the embedded video in a *.mov file format no matter what the original encoding is/was.

Topic : Mvs Internals

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Traditional Mainframes
Understand the Evolution of MVS
Understand the Traditional IBM Mainframe Operating Principles
Understand the MVS filesystem

Definition/Overview:

Multiple Virtual Storage: Multiple Virtual Storage, more commonly called MVS, was the most commonly used operating system on the System/370 and System/390 IBM mainframe computers. It was developed by IBM, but is unrelated to IBM's other mainframe operating system, VM.

Evolution of MVS: OS/MFT (Multitasking with a fixed number of Tasks) provided multitasking: several memory partitions, each of a fixed size, were set up when the

operating system was installed.

Traditional IBM Mainframe Operating Principles: Practically, it has been observed that the MVS software recovery made problem debugging both easier and more difficult: Software recovery required that programs leave 'tracks' of where they were and what they were doing, thus facilitating debugging, but the fact that processing does not stop at the time of an error, but rather progresses, can make the tracks over-written

Key Points:

1. Traditional Mainframes

Multiple Virtual Storage, more commonly called MVS, was the most commonly used operating system on the System/370 and System/390 IBM mainframe computers. It was developed by IBM, but is unrelated to IBM's other mainframe operating system, VM. First released in 1974, MVS had been renamed multiple times, first to MVS/XA (eXtended Architecture), next to MVS/ESA (Enterprise Systems Architecture), then to OS/390 (when UNIXSystem Services (USS) were added), and finally to z/OS (when 64-bit support was added with the zSeries models). Its core remains fundamentally the same operating system. By design, programs written for MVS can still run on z/OS without modification. At first IBM described MVS as simply a new release of OS/VS2. But it was in fact a complete re-write - previous OS/VS2 releases were upgrades of OS/MVT and, like MVT, were mainly written in Assembler; the core of MVS was almost entirely written in PL/S. IBM's use of "OS/VS2" emphasized upwards compatibility: application programs which ran under MVT did not even need to be re-compiled in order to run under MVS; the same Job Control Language files could be used unchanged; the utilities and other non-core facilities like TSO ran unchanged. But users almost unanimously called the new system MVS from the start, and IBM followed their lead in the naming of later major versions such as MVS/XA. After the release of MVS, users described earlier OS/VS2 releases as SVS (Single Virtual Storage).

2. Evolution of MVS

OS/MFT (Multitasking with a fixed number of Tasks) provided multitasking: several

memory partitions, each of a fixed size, were set up when the operating system was installed. For example, there might be a small partition, two medium partitions, and a large partition. If there were two large programs ready to run, one would have to wait on the other until it finished and vacated the large partition. OS/MVT (Multitasking with a Variable number of Tasks) was an enhancement which further refined memory usage. Instead of using fixed-size memory partitions, MVT allocated memory to programs as needed provided there was enough contiguous physical memory available. This was a significant advance over MFT's memory management: there was no predefined limit on the number of jobs that could run at the same time; and two or more large jobs could run at the same time if enough memory was available. But it had some weaknesses: if a job allocated memory dynamically (as most sort programs and database management systems do), the programmers had to estimate the job's maximum memory requirement and pre-define it for MVT; a job which contained a mixture of small and large programs would waste memory while the small programs were running; most seriously, memory could become fragmented, i.e. the memory not used by current jobs could be divided into uselessly small chunks between the areas used by current jobs, and the only remedy was to wait until all current jobs finished before starting any new ones. In the early 1970s IBM sought to mitigate these difficulties by introducing virtual memory (referred to by IBM as "virtual storage"), which allowed programs to request address spaces larger than physical memory. The original implementations had a single virtual address space, shared by all jobs. OS/VS1 was OS/MFT within a single virtual address space; OS/VS2 SVS was OS/MVT within a single virtual address space. So OS/VS1 and SVS in principle had the same disadvantages as MFT and MVT but the impacts were less severe because jobs could request much larger address spaces. In the mid-1970s IBM introduced MVS, which allowed an indefinite number of applications to run in different address spaces - two concurrent programs might try to access the same virtual memory address, but the virtual memory system redirected these requests to different areas of physical memory. Each of these address spaces consisted of 3 areas: operating system (one instance shared by all jobs); application area which was unique for each application; shared virtual area which was used for various purposes including inter-job communication. IBM promised that the application areas would always be at least 8MB. MVS originally supported 24-bit addressing (i.e. up to 16MB). As the

underlying hardware progressed it supported 31-bit (XA and ESA; up to 2048MB) and now (as z/OS) 64-bit addressing. Two of the most significant reasons for the rapid upgrade to 31-bit addressing were: the growth of large transaction-processing networks, mostly controlled by CICS, which ran in a single address space; the DB2 relational database management system needed more than 8MB of application address space in order to run efficiently (early versions were configured into two address spaces which communicated via the shared virtual area, but this imposed a significant overhead since all such communications had to be transmitted via the operating system).

3. Traditional IBM Mainframe Operating Principles

Practically, it has been observed that the MVS software recovery made problem debugging both easier and more difficult: Software recovery required that programs leave 'tracks' of where they were and what they were doing, thus facilitating debugging, but the fact that processing does not stop at the time of an error, but rather progresses, can make the tracks over-written. Early date capture at the time of the error maximizes debugging, and facilities exist for the recovery routines (task and system mode, both) to do this. IBM included additional criteria for a major software problem that would require IBM service to repair it: If a mainline component failed to initiate software recovery that was considered a reportable valid failure. Also, if a recovery routine failed to collect significant diagnostic data such that the original problem was solvable by data collected by that recovery routine, IBM standards dictated that this fault was reportable and required repair. Thus, IBM standards, when applied rigorously, would encourage continuous improvement. IBM introduced an on-demand hypervisor, a major serviceability tool, called Dynamic Support System (DSS), in the first release of MVS. This facility could be invoked to initiate a session to create diagnostic procedures, or invoke already-stored procedures. The procedures 'trapped' special events, such as the loading of a program, device I/O, system procedure calls, and then triggered the activation of the previously-defined procedures. These procedures, which could be invoked recursively, allowed for reading and writing of data, and alteration of instruction flow. Program Event recording hardware was used. Due to the overhead of this tool, it was removed from customer-available MVS systems. Program-Event Recording (PER) exploitation was performed by the

enhancement of the diagnostic "SLIP" command with the introduction of the PER support (SLIP/Per) in SU 64/65 (1978). Multiple copies of MVS (or other IBM operating systems) could share the same machine if that machine was controlled by VM/370 - in this case VM/370 was the real operating system and regarded the "guest" operating systems as applications with unusually high privileges. As a result of later hardware enhancements one instance of an operating system (either MVS, or VM with guests, or other) could also occupy a Logical Partition (LPAR) instead of an entire physical system. Multiple MVS instances can be organized and collectively administered in a structure called a systems complex or sysplex, introduced in September, 1990. Instances interoperate through a software component called a Cross-system Coupling Facility (XCF) and a hardware component called a Hardware Coupling Facility (CF or Integrated Coupling Facility, ICF, if co-located on the same mainframe hardware). Multiple sysplexes can be joined via standard network protocols such as IBM's proprietary Systems Network Architecture (SNA) or, more recently, via TCP/IP.

4. MVS file system

Data set names (DSNs, mainframe term for filenames) are organized in a hierarchy whose levels are separated with dots, e.g. "DEPT01.SYSTEM01.FILE01". Each level in the hierarchy can be up to eight characters long. The total filename length is a maximum of 44 characters including dots. By convention, the components separated by the dots are used to organize files similarly to directories in other operating systems. For example there were utility programs which performed similar functions to those of Windows Explorer (but without the GUI and usually in batch processing mode) - adding, renaming or deleting new elements and reporting all the contents of a specified element. However, unlike in many other systems, these levels are not actual directories but just a naming convention (like the original Macintosh File System, where folder hierarchy was an illusion maintained by the Finder). TSO supports a default prefix for files (similar to a "current directory" concept), and RACF supports setting up access controls based on filename patterns, analogous to access controls on directories on other platforms.

As with other members of the OS family, MVS' data sets were record-oriented. MVS inherited three main types from its predecessors:

Sequential data sets were normally read one record at a time from beginning to end.
In BDAM (direct access) data sets, the application program had to specify the physical location of the data it wanted to access (usually by specifying the offset from the start of the data set).
In ISAM data sets a specified section of each record was defined as a key which could be used as a key to look up specific records. The key quite often consisted of multiple fields but these had to be contiguous and in the right order; and key values had to be unique. Hence an IBM ISAM file could have only one key, equivalent to the primary key of a relational database table; ISAM could not support foreign keys.

Sequential and ISAM datasets could store either fixed-length or variable length records, or all types could occupy more than one disk volume.
All of these are based on the VTOC disk structure.
Early IBM database management systems used various combinations of ISAM and BDAM datasets - usually BDAM for the actual data storage and ISAM for indexes. These VSAM formats became the basis of IBM's database management systems, IMS/VS and DB2 - usually ESDS for the actual data storage and KSDS for indexes. VSAM also included a catalog component which was used for MVS' master catalog. Partitioned datasets (PDS) were sequential datasets which were subdivided into "members" which could be processed as sequential files in their own right. The most important use of PDS was for program libraries - system administrators used the main PDS as a way to allocate disk space to a project and the project team then created and edited the members. Generation Data Groups (GDGs) were originally designed to support grandfather-father-son backup procedures - if a file was modified, the changed version became the new "son", the previous "son" became the "father", the previous "father" became the "grandfather" and the previous "grandfather" was deleted. But one could set up GDGs with a lot more than 3 generations and some applications used GDGs to collect data from several sources and feed the information to one program - each collecting program created a new generation of the file and the final program read the whole group as a single sequential file (by not specifying a generation in the JCL). Modern versions of MVS (i.e. z/OS) also support POSIX-compatible "slash" filesystems along with facilities for integrating the two filesystems. That is, the OS can make an MVS dataset appear as a file to a POSIX program or subsystem. These newer

filesystems include Hierarchical File System (HFS) (not to be confused with Apple's Hierarchical File System) and zFS (not to be confused with Sun's ZFS).

Topic : Data Communication And Networks

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Layers
Understand the Data Communication
Understand the The Public Communication Infrastructure
Understand the Networks
Understand the ECNs
Understand the ECNs and the stock market
Understand the ECN fee structure

Definition/Overview:

A computer network is an interconnected group of computers. Networks may be classified by the network layer at which they operate according to basic reference models considered as standards in the industry, such as the five-layer Internet Protocol Suite model. While the seven-layer Open Systems Interconnection (OSI) reference model is better known in academia, the majority of networks use the Internet Protocol Suite (IP).

Key Points:

1. Layers

The Network Layer is Layer 3 (of seven) in the OSI model of networking. The Network Layer responds to service requests from the Transport Layer and issues service requests to the Data Link Layer. In essence, the Network Layer is responsible

for end-to-end (source to destination) packet delivery including any routing through intermediate hosts, whereas the Data Link Layer is responsible for node-to-node (hop-to-hop) frame delivery on the same link. The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service, and error control functions. The Network Layer deals with transmitting information all the way from its source to its destination - transmitting from anywhere, to anywhere. Here are some things that the Network Layer needs to address:

Is the network connection-oriented or connectionless?
For example, snail mail is connectionless, in that a letter can travel from a sender to a recipient without the recipient having to do anything. On the other hand, the telephone system is connection-oriented, because the other party is required to pick up the phone before communication can be established. The OSI Network Layer protocol can be either connection-oriented, or connectionless. The IP Internet Layer (equivalent to OSI's Network Layer) supports only the connectionless Internet Protocol (IP); however, connection-oriented protocols, such as TCP, exist higher in the stack by enforcing reliability constraints through timeouts and resending packets.
What are the Global Addresses?
Everybody in the network needs to have a unique address which determines who he is. This address will normally be hierarchical, so you can be "Fred Murphy" to Dubliners, or "Fred Murphy, Dublin" to people in Ireland, or "Fred Murphy, Dublin, Ireland" to people anywhere in the world. On the internet, these addresses are known as IP Addresses.
How do you forward a message?
This is of particular interest to mobile applications, where a user may rapidly move from place to place, and it must be arranged that his messages follow him. Version 4 of the Internet Protocol (IPv4) doesn't really allow for this, though it has been hacked somewhat since its inception. Fortunately, the forthcoming IPv6 has a much better designed solution, which should make this type of application much smoother.

2. Data Communication

Computer networking is the engineering discipline concerned with communication

between computer systems or devices. Networking, routers, routing protocols, and networking over the public Internet have their specifications defined in documents called RFCs. Computer networking is sometimes considered a sub-discipline of telecommunications, computer science, information technology and/or computer engineering. Computer networks rely heavily upon the theoretical and practical application of these scientific and engineering disciplines.
A computer network is any set of computers or devices connected to each other with the ability to exchange data. Examples of different networks are:

Local area network (LAN), which is usually a small network constrained to a small geographic area.
Wide area network (WAN) that is usually a larger network that covers a large geographic area.
Wireless LANs and WANs (WLAN & WWAN) are the wireless equivalent of the LAN and WAN.

All networks are interconnected to allow communication with a variety of different kinds of media, including twisted-pair copper wire cable, coaxial cable, optical fiber, and various wireless technologies. The devices can be separated by a few meters (e.g. via Bluetooth) or nearly unlimited distances (e.g. via the interconnections of the Internet).

3. The Public Communication Infrastructure

Communications infrastructure

Telephone networks (land lines) including switching systems
Mobile phone networks
Cable television networks including receiving stations and cable distribution networks
Internet backbone, including high-speed data cables, routers and servers as well as the protocols and other basic software required for the system to function
Communication satellites
Undersea cables
Major private, government or dedicated telecommunications networks, such as those used for internal communication and monitoring by major infrastructure companies,

by governments, by the military or by emergency services
Pneumatic tube mail distribution networks

4. Networks

An electronic communication network (ECN) is the term used in financial circles for a type of computer system that facilitates trading of financial products outside of stock exchanges. The primary products that are traded on ECNs are stocks and currencies. ECNs came into existence in 1998 when the SEC authorized their creation. ECNs increase competition among trading firms by lowering transaction costs, giving clients full access to their order books, and offering order matching outside of traditional exchange hours.

5. Functioning of ECNs

In order to trade with an ECN, one must be a subscriber or have an account with a broker that provides direct access trading. ECN subscribers can enter orders into the ECN via a custom computer terminal or network protocols. The ECN will then match contra-side orders (i.e. a sell-order is "contra-side" to a buy-order with the same price and share count) for execution. The ECN will post unmatched orders on the system for other subscribers to view. Generally, the buyer and seller are anonymous, with the trade execution reports listing the ECN as the party.
Some ECNs may offer additional features to subscribers such as negotiation, reserve size, and pegging, and may have access to the entire ECN book (as opposed to the "top of the book") that contains important real-time market data regarding depth of trading interest.

6. ECNs and the stock market

For stock, ECNs exist as a class of SEC-permitted Alternative Trading Systems (ATS). As an ATS, ECNs exclude broker-dealers' internal crossing networks i.e., systems that match orders at the broker-dealer using prices from an exchange, without actually sending the order to a public venue.

7. ECN fee structure

ECN's fee structure can be grouped in two basic structures: a classic structure and a credit (or rebate) structure. Both fee structures offer advantages of their own. The classic structure tends to attract liquidity removers while the credit structure appeals to

liquidity providers. However since both removers and providers of liquidity are necessary to create a market ECNs have to choose their fee structure carefully. In a credit structure ECNs make a profit from paying liquidity providers a credit while charging a debit to liquidity removers. Their fees range from $0.002 to $0.0027 per share for liquidity providers, and $0.003 to $0.0025 per share for liquidity removers. The fee can be determined by monthly volume provided and removed, or by a fixed structure, depending on the ECN, and it's known as a liquidity rebate, or credit. This structure is common on the NASDAQ market. In a classic structure, the ECN will charge a small fee to all market participants using their network, both liquidity providers and removers. They can also give lower price to large liquidity providers in order to attract volume to their networks. Fees for ECNs that operate under a classic structure range from $0 to $0.0015, or even higher depending on each ECN. This fee structure is more common in the NYSE, however recently some ECNs have moved their NYSE operations into a credit structure.

Topic : The Internet And The World Wide Web

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Internet's Infrastructure
Understand the TCP/IP, the Internet's Protocols
Understand the Layers in the Internet Protocol Suite
Understand the World Wide Web

Definition/Overview:

The World Wide Web: The World Wide Web (commonly shortened to the Web) is a system of interlinked hypertext documents accessed via the Internet. With a Web browser, a user views Web pages that may contain text, images, videos, and other multimedia and navigates between them using hyperlinks. The World Wide Web was

created in 1989 by Sir Tim Berners-Lee, working at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland and released in 1992. Since then, Berners-Lee has played an active role in guiding the development of Web standards (such as the markup languages in which Web pages are composed), and in recent years has advocated his vision of a Semantic Web.

Key Points:

1. The Internet's Infrastructure

Internet Protocol (IP)
Internet Protocol Suite (TCP/IP)
Internet service provider (ISP)

2. TCP/IP, the Internet's Protocols

The Internet Protocol Suite (commonly known as TCP/IP) is the set of communications protocols used for the Internet and other similar networks. It is named from two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which were the first two networking protocols defined in this standard. Today's IP networking represents a synthesis of several developments that began to evolve in the 1960s and 1970s, namely the Internet and LANs (Local Area Networks), which emerged in the mid- to late-1980s, together with the advent of the World Wide Web in the early 1990s. The Internet Protocol Suite, like many protocol suites, may be viewed as a set of layers. Each layer solves a set of problems involving the transmission of data, and provides a well-defined service to the upper layer protocols based on using services from some lower layers. Upper layers are logically closer to the user and deal with more abstract data, relying on lower layer protocols to translate data into forms that can eventually be physically transmitted. The TCP/IP model consists of four layers (RFC 1122). From lowest to highest, these are the Link Layer, the Internet Layer, the Transport Layer, and the Application Layer.

3. Layers in the Internet Protocol Suite

The TCP/IP suite uses encapsulation to provide abstraction of protocols and services. Such encapsulation usually is aligned with the division of the protocol suite into layers of general functionality. In general, an application (the highest level of the model) uses

a set of protocols to send its data down the layers, being further encapsulated at each level. This may be illustrated by an example network scenario, in which two Internet host computers communicate across local network boundaries constituted by their internetworking gateways (routers). The functional groups of protocols and methods are the Application Layer, the Transport Layer, the Internet Layer, and the Link Layer (RFC 1122). It should be noted that this model was not intended to be a rigid reference model into which new protocols have to fit in order to be accepted as a standard. The following table provides some examples of the protocols grouped in their respective layers.

The following table shows the layer names and the number of layers in the TCP/IP model as presented in widespread university course textbooks on computer networking used today.

	FOROUZAN	**COMER, KOZIEROK**	**STALLINGS**	**TANENBAUM**	**KUROSE, RFC 1122**	**CISCO ACADEMY**
	Five layers	Five layers	Five layers	Four layers	Four layers	Four layers
L5	Application	Application	Application	Application	Application	Application
L4	Transport	Transport	Host-to-host or transport	Transport	Transport	Transport
L3	Network	Internet	Internet	Internet	Internet	Internetwork
L2	Data link	Data link (Network interface)	Network access	Host-to-network	Link	Network interface
L1	Physical	(Hardware)	Physical			

[Table 3: These textbooks are secondary sources that may contravene the intent of RFC 1122 and other IETF primary sources]

Different authors have interpreted the RFCs differently regarding whether the **Link Layer** (and the four-layer TCP/IP model) covers physical layer issues or a "hardware

layer" is assumed below the link layer. Some authors have tried to use other names for the link layer, such as Network interface layer, in effort to avoid confusion with the Data link layer of the seven-layer OSI model. Others have attempted to map the Internet Protocol model onto the seven-layer OSI Model. The mapping often results in a five-layer TCP/IP model, wherein the Link Layer is split into a Data Link Layer on top of a Physical Layer. Especially in literature with a bottom-up approach to computer networking, where physical layer issues are emphasized, an evolution towards a five-layer Internet model can be observed out of pedagogical reasons. The Internet Layer is usually directly mapped to the OSI's Network Layer. At the top of the hierarchy, the Transport Layer is always mapped directly into OSI Layer 4 of the same name. OSIs Application Layer, Presentation Layer, and Session Layer are collapsed into TCP/IP's Application Layer. As a result, these efforts result in either a four- or five-layer scheme with a variety of layer names. This has caused considerable confusion in the application of these models. Other authors dispense with rigid pedagogy focusing instead on functionality and behavior. The Internet protocol stack has never been altered by the Internet Engineering Task Force (IETF) from the four layers defined in RFC 1122. The IETF makes no effort to follow the seven-layer OSI model and does not refer to it in standards-track protocol specifications and other architectural documents. The IETF has repeatedly stated that Internet protocol and architecture development is not intended to be OSI-compliant. RFC 3439, addressing Internet architecture, contains a section entitled: "Layering Considered Harmful".

4. The World Wide Web

The World Wide Web (commonly abbreviated as "the Web") is a system of interlinked hypertext documents accessed via the Internet. With a Web browser, one can view Web pages that may contain text, images, videos, and other multimedia and navigate between them using hyperlinks. Using concepts from earlier hypertext systems, the World Wide Web was begun in 1992 by the English physicist Tim Berners-Lee, now the Director of the World Wide Web Consortium, and Robert Cailliau, a Belgian computer scientist, while both working at CERN in Geneva, Switzerland. In 1990, they proposed building a "web of nodes" storing "hypertext pages" viewed by "browsers" on a network, and released that web in 1992. Connected by the existing Internet, other websites were created, around the world, adding

international standards for domain names & the HTML language. Since then, Berners-Lee has played an active role in guiding the development of Web standards (such as the markup languages in which Web pages are composed), and in recent years has advocated his vision of a Semantic Web. Cailliau went on early retirement in January 2005 and left CERN in January 2007. The World Wide Web enabled the spread of information over the Internet through an easy-to-use and flexible format. It thus played an important role in popularizing use of the Internet, to the extent that the World Wide Web has become a synonym for Internet, with the two being conflated in popular use. The terms Internet and World Wide Web are often used in every-day speech without much distinction. However, the Internet and the World Wide Web are not one and the same.

The Internet is a global data communications system. It is a hardware and software infrastructure that provides connectivity between computers. In contrast, the Web is one of the services communicated via the Internet. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. Viewing a Web page on the World Wide Web normally begins either by typing the URL of the page into a Web browser, or by following a hyperlink to that page or resource. The Web browser then initiates a series of communication messages, behind the scenes, in order to fetch and display it. First, the server-name portion of the URL is resolved into an IP address using the global, distributed Internet database known as the domain name system, or DNS. This IP address is necessary to contact and send data packets to the Web server. The browser then requests the resource by sending an HTTP request to the Web server at that particular address. In the case of a typical Web page, the HTML text of the page is requested first and parsed immediately by the Web browser, which will then make additional requests for images and any other files that form a part of the page. Statistics measuring a website's popularity are usually based on the number of 'page views' or associated server 'hits', or file requests, which take place. Having received the required files from the Web server, the browser then renders the page onto the screen as specified by its HTML, CSS, and other Web languages. Any images and other resources are incorporated to produce the on-screen Web page that the user sees. Most Web pages will themselves contain hyperlinks to other related pages and perhaps to downloads, source documents, definitions and other Web resources. Such a collection of useful, related resources, interconnected via hypertext links, is what was

dubbed a "web" of information. Making it available on the Internet created what Tim Berners-Lee first called the WorldWideWeb (in its original CamelCase, which was subsequently discarded) in November 1990.

In Section 5 of this course you will cover these topics:

- Client/Server Information Systems
- Windows 2003 Server
- Linux Networking.
- Novell Netware

Topic : Client/Server Information Systems

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Web Information Systems
Understand the Middleware
Understand the Use of middleware

Definition/Overview:

Web Information Systems: Web information system, or web-based information system, is an information system that utilizes Internet web technologies to deliver information and services, to users or other information systems/applications.

Middleware: Middleware is computer software that connects software components or applications.

Key Points:

1. Web Information Systems

Web information system, or web-based information system, is an information system that utilizes Internet web technologies to deliver information and services, to users or other information systems/applications. Web application is a specific functionality-oriented component that utilizes web technologies to deliver information and services to users or other applications/information systems. A web information system usually consists of one or more web applications, together with information components and other non-web components.

2. Middleware

Middleware is computer software that connects software components or applications. The software consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. This technology evolved to provide for interoperability in support of the move to coherent distributed architectures, which are used most often to support and simplify complex, distributed applications. It includes web servers, application servers, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture. Middleware sits "in the middle" between application software working on different operating systems. It is similar to the middle layer of three-tier single system architecture, except that it is stretched across multiple systems or applications. Examples include database systems, telecommunications software, transaction monitors, and messaging-and-queueing software. The distinction between operating system and middleware functionality is, to some extent, arbitrary. While core kernel functionality can only be provided by the operating system itself, some functionality previously provided by separately sold middleware is now integrated in operating systems. A typical example is the TCP/IP stack for telecommunications, nowadays

included in virtually every operating system. In simulation technology, middleware is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure. Middleware generally consists of a library of functions, and enables a number of applications simulations or federates in HLA terminology to page these functions from the common library rather than re-create them for each application IBM, Red Hat, and Oracle Corporation are major vendors providing middleware software. Vendors such as SAP, TIBCO, Mercator Software, Crossflo, Vitria and webMethods were specifically founded to provide Web-oriented middleware tools. Groups such as the Apache Software Foundation and the ObjectWeb Consortium encourage the development of open source middleware.

3. Use of middleware

Middleware services provide a more functional set of application programming interfaces to allow an application to:

Locate transparently across the network, thus providing interaction with another service or application

Be independent from network services

Be reliable and available always when compared to the operating system and network services.

Topic : Windows 2003 Server

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Windows Server 2003

Understand the Windows 2003 Network Architecture

Understand the File Services

Understand the Print Services

Understand the Web Services

Understand the Media Services
Understand the Clustering Services
Understand the Windows HPC Server 2008
Understand the Windows Storage Server

Definition/Overview:

Windows Server 2003 : Windows Server 2003 (also referred to as Win2K3) is a server operating system produced by Microsoft. Introduced on 24 April 2003 as the successor to Windows 2000 Server, it is considered by Microsoft to be the cornerstone of its Windows Server System line of business server products.

The Windows 2003 Network Architecture: SBS includes Windows Server and additional technologies aimed at providing a small business with a complete technology solution.

File Services: Most editions of Windows Server 2008 are available in x86-64 (64-bit) and x86 (32-bit) versions.

Print Services: A print server, or printer server, is a computer or device that is connected to one or more printers and to client computers over a network, and can accept print jobs from the computers and send the jobs to the appropriate printers.

Web Services: Windows Server 2003, Web Edition is mainly for building and hosting Web applications, Web pages, and XML Web services. It is designed to be used primarily as an IIS 6.0 Web server and provides a platform for rapidly developing and deploying XML Web services and applications that use ASP.NET technology, a key part of the .NET Framework.

Key Points:

1. Windows Server 2003
Windows Server 2003 (also referred to as Win2K3) is a server operating system produced by Microsoft. Introduced on 24 April 2003 as the successor to Windows 2000 Server, it is considered by Microsoft to be the cornerstone of its Windows Server System line of business server products. An updated version, Windows Server 2003

R2 was released to manufacturing on 6 December 2005. Its successor, Windows Server 2008, was released on 4 February 2008. According to Microsoft, Windows Server 2003 is more scalable and delivers better performance than its predecessor, Windows 2000.

2. The Windows 2003 Network Architecture

SBS includes Windows Server and additional technologies aimed at providing a small business with a complete technology solution. The technologies are integrated to enable small business with targeted solutions such as the Remote Web Workplace, and offer management benefits such as integrated setup, enhanced monitoring, a unified management console, and remote access.

The Standard Edition of SBS includes Windows SharePoint Services for collaboration, Microsoft Exchange server for e-mail, Fax Server, and the Active Directory for user management. The product also provides a basic firewall, DHCP server and NAT router using either two network cards or one network card in addition to a hardware router.

The Premium Edition of SBS includes the above plus Microsoft SQL Server 2000 and Microsoft Internet Security and Acceleration Server 2004.

SBS has its own type of Client Access License (CAL) that is different and costs slightly more than CALs for the other editions of Windows Server 2003. However, the SBS CAL encompasses the user CALs for Windows Server, Exchange Server, SQL Server, and ISA Server, and hence is less expensive than buying all the other CALs individually.

SBS server has the following design limitations:

Only one computer in a domain can be running Windows Server 2003 for Small Business Server.

Windows Server 2003 for Small Business Server must be the root of the Active Directory forest.

Windows Server 2003 for Small Business Server cannot trust any other domains.

Windows Server 2003 for Small Business Server is limited to 75 users or devices depending on which type of CAL.

Windows Server 2003 for Small Business Server is limited to 4GB of RAM (Random

Access Memory).
A Windows Server 2003 for Small Business Server domain cannot have any child domains.
Terminal Services only operates in remote administration mode on the server running SBS 2003, and only two simultaneous RDP sessions are allowed. (Change from SBS 2000 policy)
To remove the limits from SBS server and upgrade from Small Business Server to regular Windows Server, Exchange Server, SQL and ISA server versions there is a Windows Small Business Server 2003 R2 Transition Pack.

3. File Services

Most editions of Windows Server 2008 are available in x86-64 (64-bit) and x86 (32-bit) versions. Windows Server 2008 for Itanium-based Systems supports IA-64 processors. The IA-64 version is optimized for high workload scenarios like database servers and Line of Business (LOB) applications. As such it is not optimized for use as a file server or media server. Microsoft has announced that Windows Server 2008 is the last 32-bit Windows server operating system. Windows Server 2008 is available in the editions listed below,similar to Windows Server 2003.

Windows Server 2008 Standard Edition (x86 and x64)
Windows Server 2008 Enterprise Edition (x86 and x64)
Windows Server 2008 Datacenter Edition (x86 and x64)
Windows HPC Server 2008 (replacing Windows Compute Cluster Server 2003)
Windows Web Server 2008 (x86 and x64)
Windows Storage Server 2008 (x86 and x64)
Windows Small Business Server 2008 (Codenamed "Cougar") (x64) for small businesses
Windows Essential Business Server 2008 (Codenamed "Centro") (x64) for medium-sized businesses
Windows Server 2008 for Itanium-based Systems

Server Core is available in the Standard, Enterpriseand Datacenter editions. It is not available in Web edition or in the Itanium edition. Server Core is simply a server role

supported by some of the editions, and not a separate edition by itself. Each architecture has a separate installation DVD. Windows Server 2008 Standard Edition is available to students for free through Microsoft's DreamSpark program.

4. Print Services

A print server, or printer server, is a computer or device that is connected to one or more printers and to client computers over a network, and can accept print jobs from the computers and send the jobs to the appropriate printers.
The term can refer to:

A host computer running Windows OS with one or more shared printers. Client computers connect using Microsoft Network Printing protocol.
A computer running some other operating system, but still implementing the Microsoft Network Printing protocol (typically Samba running on a UNIXor Linux computer).
A computer that implements the Line Printer Daemon protocol and thus can process print requests from LPD clients.
A dedicated device that connects one or more printers to a local area network (LAN). It typically has a single LAN connector, such as an RJ-45 socket, and one or more physical ports (e.g. serial, parallel or USB (Universal Serial Bus)) to provide connections to printers. In essence this dedicated device provides printing protocol conversion from what was sent by client computers to what will be accepted by the printer. Dedicated print server devices may support a variety of printing protocols including LPD/LPR over TCP/IP, NetWare, NetBIOS/NetBEUI over NBF, TCPPort 9100 or RAW printer protocol over TCP/IP, DLC or IPX/SPX. Dedicated server appliances tend to be fairly simple in both configuration and features. However these are available integrated with other devices such as a wireless router, a firewall, or both.
A dedicated device similar to (4) above, that also implements Microsoft Networking protocols to appear to Windows client computers as if it were a print server defined in (1).

5. Web Services

Windows Server 2003, Web Edition is mainly for building and hosting Web applications, Web pages, and XML Web services. It is designed to be used primarily as an IIS 6.0 Web server and provides a platform for rapidly developing and deploying XML Web services and applications that use ASP.NET technology, a key part of the .NET Framework. This edition does not require Client Access Licenses and Terminal Server mode is not included on Web Edition. However, Remote Desktop for Administration is available on Windows Server 2003, Web Edition. Only 10 concurrent file-sharing connections are allowed at any moment. It is not possible to install Microsoft SQL Server and Microsoft Exchange software in this edition. However MSDE and SQL Server 2005 Express are fully supported after service pack 1 is installed. Despite supporting XML Web services and ASP.NET, UDDI cannot be deployed on Windows Server 2003, Web Edition. The .NET Framework version 2.0 is not included with Windows Server 2003, Web Edition, but can be installed as a separate update from Windows Update.

Windows Server 2003 Web Edition supports a maximum of 2 processors with support for a maximum of 2GB of RAM. Additionally, Windows Server 2003, Web Edition cannot act as a domain controller. Additionally, it is the only version of Windows Server 2003 that does not include client number limitation upon Windows update services as it does not require Client Access Licenses.

6. Media Services

Windows Media Player (abbreviated WMP) is a digital media player and media library application developed by Microsoft that is used for playing audio, video and viewing images on personal computers running the Microsoft Windows operating system, as well as on Pocket PC and Windows Mobile-based devices. Editions of Windows Media Player were also released for Mac OS, Mac OS X and Solaris but development of these has since been discontinued. In addition to being a media player, Windows Media Player includes the ability to rip music from and copy music to compact discs, build Audio CDs in recordable discs and synchronize content with a

digital audio player (MP3 player) or other mobile devices, and enables users to purchase or rent music from a number of online music stores. Windows Media Player replaced an earlier piece of software called Media Player, adding features beyond simple video or audio playback. The current version, Windows Media Player 11, was released on October 30, 2006. Its successor, Windows Media Player 12, is under development; an initial test version was demonstrated in October 2008 as part of Windows 7. The default file formats are Windows Media Video (WMV), Windows Media Audio (WMA), and Advanced Systems Format (ASF), and supports its own XML based playlist format called Windows Playlist (WPL). The player is also able to utilize a digital rights management service in the form of Windows Media DRM.

7. Clustering Services

Windows Compute Cluster Server 2003 (CCS), released in June 2006, is designed for high-end applications that require high performance computing clusters. It is designed to be deployed on numerous computers to be clustered together to achieve supercomputing speeds. Each Compute Cluster Server network comprises at least one controlling head node and subordinate processing nodes that carry out most of the work. Computer Cluster Server uses the Microsoft Messaging Passing Interface v2 (MS-MPI) to communicate between the processing nodes on the cluster network. It ties nodes together with a powerful inter-process communication mechanism which can be complex because of communications between hundreds or even thousands of processors working in parallel. The application programming interface consists of over 160 functions. A job launcher enables users to execute jobs to be executed in the computing cluster. MS MPI was designed to be compatible with the reference open source MPI2 specification which is widely used in High-performance computing (HPC). With some exceptions because of security considerations, MS MPI covers the complete set of MPI2 functionality as implemented in MPICH2, except for the planned future features of dynamic process spawn and publishing.

8. Windows HPC Server 2008

Windows HPC Server 2008, released in September 2008, is the successor product to Windows Compute Cluster Server 2003. Like WCCS, Windows HPC Server 2008 is designed for high-end applications that require high performance computing clusters. This version of the server efficiently scales to thousands of cores and includes features

unique to HPC workloads: a new high-speed NetworkDirect RDMA, highly efficient and scalable cluster management tools, a service-oriented-architecture (SOA) job scheduler, and cluster interoperability through standards such as the High Performance Computing Basic Profile (HPCBP) specification produced by the Open Grid Forum (OGF).

9. Windows Storage Server

Windows Storage Server 2003, a part of the Windows Server 2003 series is a specialized server Operating System for Network Attached Storage (NAS). It is optimized for use in file and print sharing and also in Storage Area Network (SAN) scenarios. It is only available through Original equipment manufacturers (OEMs). Unlike other Windows Server 2003 editions that provide file and printer sharing functionality, Windows Storage Server 2003 does not require any Client access licenses. Windows Storage Server 2003 NAS equipment can be headless, which means that they are without any monitors, keyboards or mice, and are administered remotely. Such devices are plugged into any existing IP network and the storage capacity is available to all users. Windows Storage Server 2003 can use RAID arrays to provide data redundancy, fault-tolerance and high-performance. Multiple such NAS servers can be clustered to appear as a single device. This allows for very high performance as well as allowing the service to remain up even if one of the servers goes down.

10. Peer-to-Peer Networks

Peer Name Resolution Protocol (PNRP) is a patented peer-to-peer protocol designed by Microsoft. PNRP enables dynamic name publication and resolution, and requires IPv6. PNRP was first mentioned during a presentation at a P2P conference in November 2001. It appeared in July 2003 in the Advanced Networking Pack for Windows XP, and was later included in the Service Pack 2 for Windows XP. PNRP 2.0 was introduced with Windows Vista and is available for download for Windows XP Service Pack 2 users. PNRP 2.1 is included in Windows Vista SP1, Windows Server 2008 and Windows XP SP3. The design of PNRP is covered by US Patent #7,065,587, issued on June 20, 2006. PNRP is a distributed name resolution protocol allowing Internet hosts to publish "peer names" and corresponding IPv6 addresses and

optionally other information. Other hosts can then resolve the peer name, retrieve the corresponding addresses and other information, and establish peer-to-peer connections. With PNRP, peer names are composed of an "authority" and a "qualifier". The authority is identified by a secure hash of an associated public key, or by a simple place-holder (the number zero) if the peer name is "unsecured". The qualifier is a simple string, allowing an authority to have different peer names for different services. If a peer name is secure, the PNRP name records are signed by the publishing authority, and can be verified using its public key. Unsecured peer names can be published by anybody, without possible verification. Multiple entities can publish the same peer name. For example, if a peer name is associated with a group, any group member can publish addresses for the peer name. Peer names are published and resolved within a specified scope. The scope can be a local link, a site (e.g. a campus), or the whole Internet.

11. Managing Windows 2003 Server

Windows Server 2003 R2, an update of Windows Server 2003, was released to manufacturing on 6 December 2005. It is distributed on two CDs, with one CD being the Windows Server 2003 SP1 CD. The other CD adds many optionally installable features for Windows Server 2003. The R2 update was released for all x86 and x64 versions, but not for Itanium versions.

Topic : Linux Networking.

Topic Objective:

At the end of this topic student will be able to understand:

Understand the Linux File Services
Understand the Advantages
Understand the Windows Interconnectivity

Understand the Apache Web Server
Understand the Features

Definition/Overview:

Linux File Services: The ext3 or third extended filesystem is a journaled file system that is commonly used by the Linux operating system

Windows Interconnectivity: Although there is a lack of Linux ports for some Mac OS X and Microsoft Windows programs in domains such as desktop publishing and professional audio, support for common applications roughly equivalent to those available for Mac and Windows is available for Linux.

Apache Web Server: The Apache HTTP Server is a web server notable for playing a key role in the initial growth of the World Wide Web. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance.

Key Points:

1. Introduction:

Wicd, which stands for Wireless Interface Connection Daemon, is an open source software utility to manage both wireless and wired networks for Linux. The project started in late 2006 with the creation of Connection Manager, which eventually became Wicd. Wicd aims to provide a simple interface to connect to networks with a wide variety of settings. Wicd will only automatically connect to wireless networks you have specified and will not automatically connect to an unknown network. Wicd supports a unique type of system for wireless encryption using the powerful wpa_supplicant. Users can design their own "templates", which can be used by Wicd to connect to a large variety of networks using any type of encryption wpa_supplicant supports. Because Wicd does not use any GNOME dependencies, it can easily be used on GNOME, KDE, Xfce, Fluxbox, or any other window manager. Wicd is split into two major components: the daemon, and the user interface. These two components

communicate via D-Bus. This design allows the user interface to run as a standard user, and the daemon to run as the root user, so the user can change the wireless network without knowing the root password. The split interface/daemon design would also allow a person to write a new front-end to the Wicd daemon, such as wicd-qt Wicd is currently included in some distributions, such as Slackware, Zenwalk Linux and Arch Linux . Wicd has also seen some use on the EEE PC .

2. Linux File Services

The ext3 or third extended filesystem is a journaled file system that is commonly used by the Linux operating system. It is the default file system for many popular Linux distributions. Stephen Tweedie first revealed that he was working on extending ext2 in Journaling the Linux ext2fs Filesystem in 1998 paper and later in a February 1999 kernel mailing list posting and the filesystem was merged with the mainline Linux kernel in November 2001 from 2.4.15 onward. Its main advantage over ext2 is journaling which improves reliability and eliminates the need to check the file system after an unclean shutdown.

3. Advantages

Although its performance (speed) is less attractive than competing Linux filesystems such as JFS, ReiserFS and XFS, it has a significant advantage in that it allows in-place upgrades from the ext2 file system without having to back up and restore data. Ext3 also uses less CPU power than ReiserFS and XFS. It is also considered safer than the other Linux file systems due to its relative simplicity and wider testing base.
The ext3 file system adds, over its predecessor:

A Journaling file system
Online file system growth
htree indexing for larger directories (specialized version of a B-tree not to be confused with the H tree fractal)

Without these, any ext3 file system is also a valid ext2 file system. This has allowed well-tested and mature file system maintenance utilities for maintaining and repairing

ext2 file systems to also be used with ext3 without major changes. The ext2 and ext3 file systems share the same standard set of utilities, e2fsprogs, which includes a fsck tool. The close relationship also makes conversion between the two file systems (both forward to ext3 and backward to ext2) straightforward.
While in some contexts the lack of "modern" filesystem features such as dynamic inode allocation and extents could be considered a disadvantage, in terms of recoverability this gives ext3 a significant advantage over file systems with those features. The file system metadata is all in fixed, well-known locations, and there is some redundancy inherent in the data structures that may allow ext2 and ext3 to be recoverable in the face of significant data corruption, where tree-based file systems may not be recoverable.

4. Windows Interconnectivity

Although there is a lack of Linux ports for some Mac OS X and Microsoft Windows programs in domains such as desktop publishing and professional audio, support for common applications roughly equivalent to those available for Mac and Windows is available for Linux. Most Linux distributions provide a program for browsing a list of thousands of free software applications that have already been tested and configured for a specific distribution. These free programs can be downloaded and installed with one mouse click. A digital signature guarantees that no one has added a virus or a spyware to these programs. The two main frameworks for developing graphical applications are those of GNOME and KDE. These projects are based on the GTK+ and Qt widget toolkits, respectively, which can also be used independently of the larger framework. Both support a wide variety of languages. Many free software titles that are popular on Windows, such as Pidgin, Mozilla Firefox, OpenOffice.org, and GIMP, are also available in versions that run on Linux. A growing amount of proprietary desktop software is also supported, see List of proprietary software for Linux. In the field of animation and visual effects, most high end software, such as AutoDesk Maya, Softimage XSI and Apple Shake, is available for Linux, Windows and/or Mac OS X. CrossOver is a proprietary solution based on the open source Wine project that supports running Windows versions of Microsoft Office, Intuit applications such as Quicken and QuickBooks, Adobe Photoshop versions through CS2, and many popular games such as World of Warcraft and Team Fortress 2.

Besides the free Windows compatibility layer Wine, most distributions offer dual boot and x86 virtualization for running both Linux and Windows on the same computer. According to the Wine developers, "Wine is still under development, and it is not yet suitable for general use." The collaborative nature of free software development allows distributed teams to localize Linux distributions for use in locales where localizing proprietary systems would not be cost-effective. For example the Sinhalese language version of the Knoppix distribution was available for a long time before Microsoft Windows XP was translated to Sinhalese. In this case the Lanka Linux User Group played a major part in developing the localized system by combining the knowledge of university professors, linguists, and local developers. The performance of Linux on the desktop has been a controversial topic; for example, Con Kolivas accused the Linux community of favoring performance on servers. He quit Linux kernel development because he was frustrated with this lack of focus on the desktop, and then gave a "tell all" interview on the topic.

5. Apache Web Server

The Apache HTTP Server, is a web server notable for playing a key role in the initial growth of the World Wide Web. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance. The majority of all web servers using Apache are Linux web servers. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including UNIX, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is characterized as free software and open source software. Since April 1996 Apache has been the most popular HTTP server on the World Wide Web. As of December 2008 Apache served over 51% of all websites.

6. Features

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces

support mod_perl, mod_python, Tcl, and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include SSL and TLS support (mod_ssl), a proxy module, a URL rewriter (also known as a rewrite engine, implemented under mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter). Popular compression methods on Apache include the external extension module, mod_gzip, implemented to help with reduction of the size (weight) of web pages served over HTTP. Apache logs can be analyzed through a web browser using free scripts such as AWStats/W3Perl or Visitors. Virtual hosting allows one Apache installation to serve many different actual websites. For example, one machine, with one Apache installation could simultaneously serve www.example.com, www.test.com, test47.test-server.test.com, etc. Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs). Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides. Apache is the web server component of the popular LAMP web server application stack, alongside MySQL, and the PHP/Perl/Python (and now also Ruby) programming languages. Apache is redistributed as part of various proprietary software packages including the Oracle Database or the IBM WebSphere application server. Mac OS X integrates Apache as its built-in web server and as support for its WebObjects application server. It is also supported in some way by Borland in the Kylix and Delphidevelopment tools. Apache is included with Novell NetWare 6.5, where it is the default web server. Apache is also included with many Linux distributions. Apache is used for many other tasks where content needs to be made available in a secure and reliable way. One example is sharing files from a personal computer over the Internet. A user who has Apache installed on their desktop can put arbitrary files in the Apache's document root which can then be shared. Programmers developing web applications often use a locally installed version of Apache in order to preview and test code as it is being developed. Microsoft Internet Information Services (IIS) is the main competitor to Apache, trailed by Sun Microsystems' Sun Java System Web Server and a host of other applications such as Zeus Web Server. Some of the biggest web sites in the world are run using Apache. Google's search engine front end

is based on a modified version of Apache, named Google Web Server (GWS). Several Wikimedia projects also run on Apache servers.

Topic : Novell Netware

Topic Objective:

At the end of this topic student will be able to understand:

Understand the NetWare
Understand the The rise of NetWare
Understand the File Systems
Understand the Printing Services
Understand the Novell Directory Services
Understand the Features
Understand the Network Management

Definition/Overview:

NetWare: NetWare is a network operating system developed by Novell, Inc. It initially used cooperative multitasking to run various services on a PC, and the network protocols were based on the archetypal Xerox XNS stack.

The rise of NetWare: The popular use and growth of Novell NetWare began in 1985 with the simultaneous release of NetWare 286 2.0a and the Intel 80286 16-bit processor

File Systems: At the time NetWare was first developed, nearly all LAN storage was based on the disk server model.

Printing Services: A print server, or printer server, is a computer or device that is connected to one or more printers and to client computers over a network, and can accept print jobs from the computers and send the jobs to the appropriate printers.

Key Points:

1. NetWare

NetWare is a network operating system developed by Novell, Inc. It initially used cooperative multitasking to run various services on a PC, and the network protocols were based on the archetypal Xerox Network Services stack. NetWare has been superseded by Open Enterprise Server (OES). The latest version of NetWare is v6.5 Support Pack 8, which is identical to OES 2 SP1, NetWare Kernel.

2. The rise of NetWare

The popular use and growth of Novell NetWare began in 1985 with the simultaneous release of NetWare 286 2.0a and the Intel 80286 16-bit processor. The 80286 CPU featured a new 16-bit protected mode that provided access to up to 16 MB RAM as well as new mechanisms to aid multi-tasking. Prior to the 80286 CPU servers were based on the Intel 8086/8088 8/16-bit processors, which were limited to an address space of 1MB with not more than 640 KB of directly addressable RAM. The combination of a higher 16 MB RAM limit, 80286 processor feature utilization, and 256 MB NetWare volume size limit allowed reliable, cost-effective server-based local area networks to be built for the first time. The 16 MB RAM limit was especially important, since it made enough RAM available for disk caching to significantly improve performance. This became the key to Novell's performance while also allowing larger networks to be built. Another significant difference of NetWare 286 was that it was hardware-independent, unlike competing server systems from 3Com.

Novell servers could be assembled using any brand system with an Intel 80286 or higher CPU, any MFM, RLL, ESDI, or SCSI hard drive and any 8- or 16-bit network adapter for which Netware drivers were available. Novell also designed a compact and simple DOS client software program that allowed DOS stations to connect to a server and access the shared server hard drive. While the NetWare server file system introduced a new, proprietary file system design, it looked like a standard DOS volume to the workstation, ensuring compatibility with all existing DOS programs.

3. File Systems

At the time NetWare was first developed, nearly all LAN storage was based on the disk server model. This meant that if a client computer wanted to read a particular block from a particular file it would have to issue the following requests across the relatively slow LAN:

Read first block of directory
Continue reading subsequent directory blocks until the directory block containing the information on the desired file was found, could be many directory blocks
Read through multiple file entry blocks until the block containing the location of the desired file block was found, could be many directory blocks
Read the desired data block
NetWare, since it was based on a file service model, interacted with the client at the file API level:
Send file open request (if this hadn't already been done)
Send a request for the desired data from the file
All of the work of searching the directory to figure out where the desired data was physically located on the disk was performed at high speed locally on the server. By the mid-1980s, most NOS products had shifted from the disk service to the file service model. Today, the disk service model is making a comeback, see SAN.

4. Printing Services

A print server, or printer server, is a computer or device that is connected to one or more printers and to client computers over a network, and can accept print jobs from the computers and send the jobs to the appropriate printers.

The term can refer to:

A host computer running Windows OS with one or more shared printers. Client computers connect using Microsoft Network Printing protocol.
A computer running some other operating system, but still implementing the Microsoft Network Printing protocol (typically Samba running on a UNIXor Linux computer).
A computer that implements the Line Printer Daemon protocol and thus can process print requests from LPD clients.
A dedicated device that connects one or more printers to a local area network (LAN). It typically has a single LAN connector, such as an RJ-45 socket, and one or more physical ports (e.g. serial, parallel or USB (Universal Serial Bus)) to provide connections to printers. In essence this dedicated device provides printing protocol conversion from what was sent by client computers to what will be accepted by the printer. Dedicated print server devices may support a variety of printing protocols including LPD/LPR over TCP/IP, NetWare, NetBIOS/NetBEUI over NBF, TCPPort 9100 or RAW printer protocol over TCP/IP, DLC or IPX/SPX. Dedicated server appliances tend to be fairly simple in both configuration and features. However these are available integrated with other devices such as a wireless router, a firewall, or both.
A dedicated device similar to (4) above, that also implements Microsoft Networking protocols to appear to Windows client computers as if it were a print server defined in (1).

The term print server usually refers to (1) or (2) above, while print server device or print server appliance usually refers to (4).

5. Novell Directory Services

Novell eDirectory (formerly called NetWare Directory Services, NDS) is an X.500 compatible directory service software product released in 1993 by Novell, Inc. for centrally managing access to resources on multiple servers and computers within a given network.

6. Features

Novell Directory Services is a hierarchical, object oriented database that represents all the assets in an organization in a logical tree. Assets can include people, positions, servers, workstations, applications, printers, services, groups, etc. The use of dynamic rights inheritance and equivalence allows both global and fine grained access controls to be implemented efficiently. Access rights between objects in the tree are determined at the time of the request and are determined by the rights assigned to the objects by virtue of their location in the tree, any security equivalences and individual assignments. eDirectory supports partitioning at any point in the tree and replication of that partition to any number of servers. Replication between each server occurs periodically using deltas of the objects to reduce LAN/WAN traffic. Each server can act as a master of the information it holds (providing the replica is not read only). Additionally, replicas may be filtered to only include defined attributes to increase speed (eg a replica may be configured to only include a user's name and phone number for use in a corporate address book). eDirectory supports referential integrity, multi-master replication and has a modular authentication architecture. It can be accessed via LDAP, DSML, SOAP, ODBC, JDBC, JNDI and ADSI and has been proven to scale to over 1 billion objects.

7. Network Management

One of the raging debates of the 90s was whether it was more appropriate for network file service to be performed by a software layer running on top of a general purpose operating system, or by a special purpose operating system. NetWare was a special purpose operating system, not a timesharing OS. It was written from the ground up as a platform for client-server processing services. Initially it focused on file and print services, but later demonstrated its flexibility by running database, email, web and other services as well. It also performed efficiently as a router, supporting IPX, TCP/IP, and Appletalk, though it never offered the flexibility of a 'hardware' router. In 4.x and earlier versions, NetWare did not support preemption, virtual memory, graphical user interfaces, etc. Processes and services running under the NetWare OS were expected to be cooperative that is to process a request and return control to the OS in a timely fashion. On the down side, this trust of application processes to manage themselves could lead to a misbehaving application bringing down the server. By comparison, general purpose operating systems such as UNIXor Microsoft Windows

were based on an interactive, time-sharing model where competing programs would consume all available resources if not held in check by the Operating System. Such environments operated by preemption, memory virtualization, etc., generating significant overhead because there were never enough resources to do everything every application desired. These systems improved over time as network services shed their application stigma and moved deeper into the kernel of the general purpose OS, but they never equaled the efficiency of NetWare. Probably the single greatest reason for Novell's success during the 80's and 90's was the efficiency of NetWare compared to general purpose operating systems. However, as microprocessors increased in power, efficiency became less and less of an issue. With the introduction of the Pentium processor, NetWare's performance advantage began to be outweighed by the complexity of managing and developing applications for the NetWare environment.

8. NetWare 6.5

NetWare 6.5 was released in August 2003. Some of the new features in this version were:

more open-source products such as PHP, MySQL and OpenSSH
a port of the Bash shell and a lot of traditional UNIXutilities such as wget, grep, awk and sed to provide additional capabilities for scripting
iSCSI support (both target and initiator)
Virtual Office - an "out of the box" web portal for end users providing access to e-mail, personal file storage, company address book, etc.
Domain controller functionality
Universal password
DirXML Starter Pack - synchronization of user accounts with another eDirectory tree, a Windows NT domain or Active Directory.
exteNd Application Server - a J2EE 1.3-compatible application server
support for customized printer driver profiles and printer usage auditing
NX bit support
support for USB storage devices
support for encrypted volumes

9. Using NetWare

Netware Lite / Personal Netware

In 1991 Novell introduced a radically different and cheaper product - Netware Litein answer to Artisoft's similar LANtastic. Both were peer to peer systems, where no specialist server was required, but instead all PCs on the network could share their resources. The product line became Personal Netware in 1993.