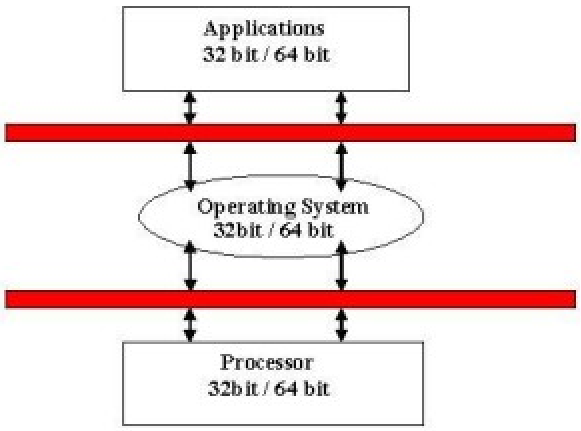
What is Processor Mode?

Processor Mode also called as CPU modes or CPU "privilege level". The CPU modes are used by processor to create an operating environment for automatic. Specifically, the CPU mode controls how the processor sees and manages the system memory and task that use it. There are three different modes of operation but one more mode is added for new 64 bit processor:

1. Real Mode.
2. Protected Mode.
3. Virtual Mode Real Mode.
4. 64 bit extension Mode.

Real Mode:

The original IBM PC could only address 1MB of system memory and the original versions of DOS created to work on it were designed with this in mind. DOS is it's by nature a single tasking operating system meaning it can only handle one program running at a time. The decision made in this early days have carried forward until now and in each new processor care had to be taken to be able to put the processor in a mode that would be compatible with the original Intel 8088 chip. This is called Real Mode. Real mode is of course used by DOS and "standard" DOS application.



Protected Mode:

Starting with the 80286 chip in the IBM AT, a new CPU mode was published called protected mode. This is much more powerful mode of operation than real mode and is used in all modern multitasking operating systems. The advantages of protected mode (compared to real mode) are:

- Full access to all of the system memory. There is no1 MB limitation in mode
- Ability to multiple tasking meaning having the operating system manages the execution of multiple programs simultaneously.
- Preference to for virtual memory which gives the permission the system to use the hard disk to emulate additional system memory when needed.
- Faster (32 bit) access to memory and faster 32 bit drivers to do I/O transfer.

The name of this mode comes from its primary use which is multitasking operating system. Each program that is running has its own assigned memory location which is protected from conflict with other programs. If a program tryes to access a memory address that it isn't allowed to a "protection fault" is generated.

Protected mode is now used by the most people use their PCs.

Virtual Mode:
This mode is also called virtual 8086 mode. The third mode of operation is actually an some more capability, an enhancement of protected mode. The use Protected mode is for to run graphical multitasking operating system such as the various types of windows. There is often desire to be able to run DOS program under the window, but DOS programs need to be run in the real mode not protected mode. Virtual real mode is used to solve this problem. Virtual real mode is also used when you use a DOS box or run a DOS game in Windows 95.

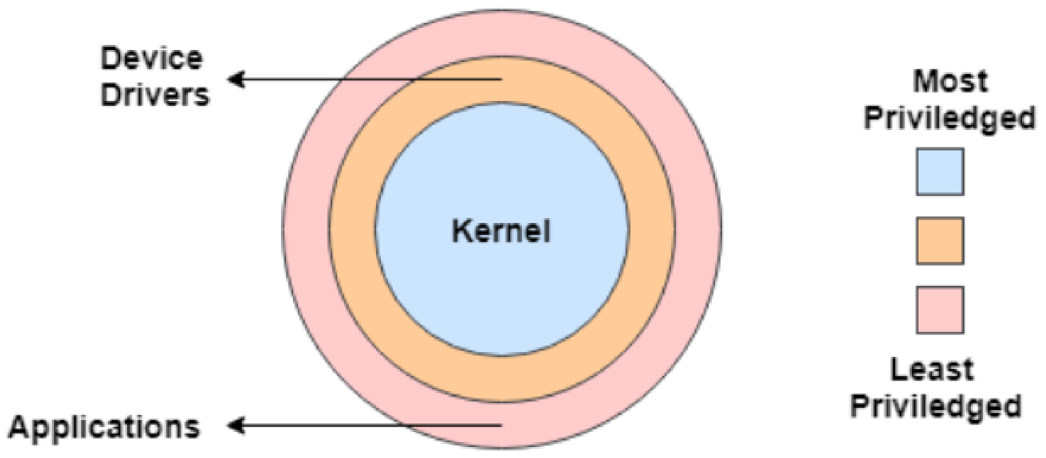
64 Bit Extension Mode:
This mode is also called Long Mode. 64 bit extension mode is the mode where a 64 bit application can access the 64 bit instruction and registers while 32 bit and 16 bit programs are executed in a compatibility sub mode. This is what about the processor modes.

For more reading about technology news in singapore and seo to online marketing do view more about other pages.

Supervisor Mode (Privileged Mode)

Supervisor mode or privileged mode is a computer system mode in which all instructions such as privileged instructions can be performed by the processor. Some of these privileged instructions are interrupt instructions, input output management etc

The privilege levels of different components in a system is given as follows:



The kernel is the most privileged part of the computer system. There are some privileged instructions that can only be executed in kernel mode or supervisor mode. The the privilege reduces for device drivers and applications respectively.

Features of Supervisor Mode

Some of the important features of supervisor mode are as follows:

- Supervisor mode handles different types of commands but mostly deals with privileged instructions. This mode is used by the operating system and has full access to all the system components.
- The system starts in supervisor mode when it boots. This allows various programs complete access to the system hardware such as bootloader, BIOS, operating system etc.
- The operating system selects supervisor mode for the low level tasks that require complete access to the system hardware.
- Supervisor mode provides the essential barrier between applications and system hardware .It also provides access to various peripherals, memory management hardware etc.
- Supervisor mode can create memory address spaces as well as update them. It can also access the memory address spaces of other operations.
- Various interrupts can be enabled or disabled using the supervisor mode. It also contributes to the loading of the processor status.
- The supervisor mode can access the various data structures available inside the operating system.

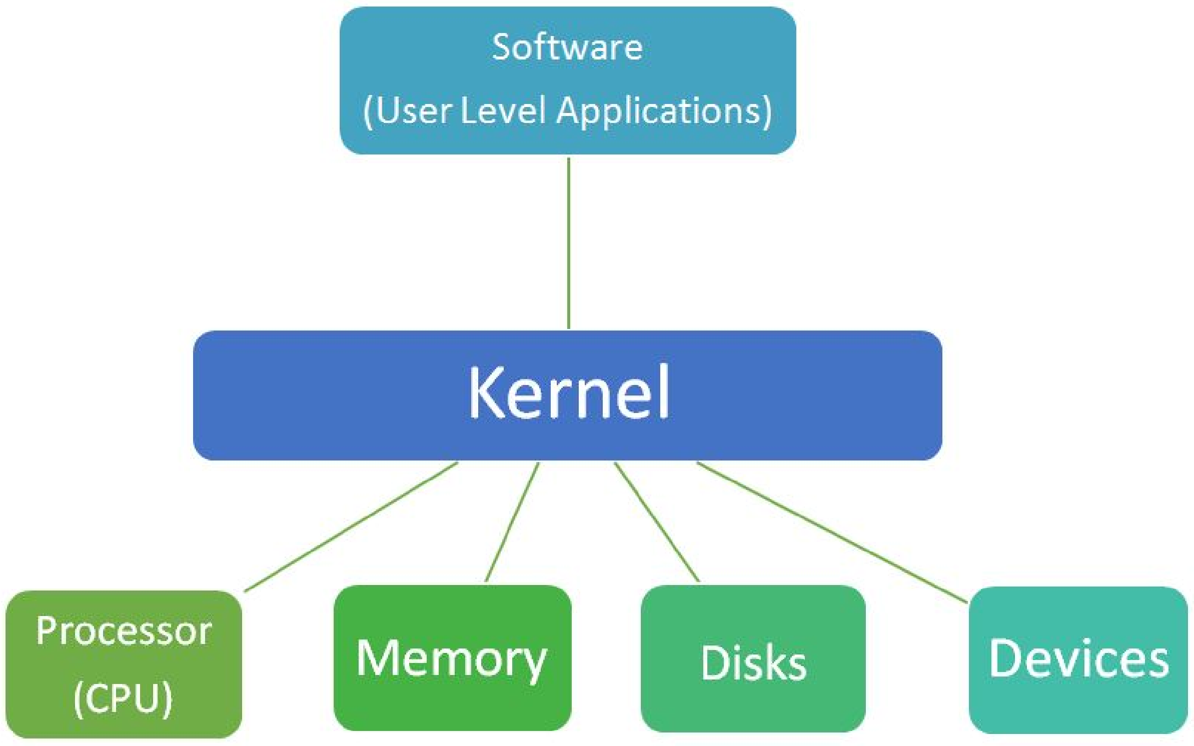
Necessity of Supervisor Mode in Operating System

The lack of a supervisor mode in an operating system can cause serious problems. Some of these are:

- A running user program can accidentally wipe out the operating system by overwriting it with user data.
- Multiple processes can write in the same system at the same time, with disastrous results.

Kernel in Operating System

A Kernel is the central component of an Operating System. The **Kernel** is also said to be the heart of the Operating System. It is responsible for managing all the processes, memory, files, etc. The **Kernel** functions at the lowest level of the Operating System. It acts as an interface (bridge) between the user-level application (software) and the hardware. Therefore, the communication between the software and the hardware is done via the Kernel.



The main functions that the Kernel performs are as follows:

1. Process Management
2. Memory Management
3. Device Management
4. Interrupt Handling
5. Input Output Communication

Now let us understand these functions of Kernel...

Functions of the Kernel in Operating System

1) Process Management

The creation, execution, and termination of processes keep on going inside the system whenever a system is in the ON mode. A process contains all the information about the task that needs to be done. So, for executing any task, a process is created inside the systems. At a time, there are many processes which are in live state inside the system. The management of all these processes is very important to avoid deadlocks and for the proper functioning of the system, and it is handled by the Kernel.

2) Memory management

Whenever a process is created and executed, it occupies memory, and when it gets terminated, the memory can be used again. But the memory should be handled by someone so that the released memory can be assigned again to the new processes. This task is also done by the Kernel. The kernel keeps track about which part of the memory is currently allocated and which part is available for being allocated to the other processes.

3) Device Management

The Kernel also manages all the different devices which are connected to the system, like the Input and Output devices, etc.

4) Interrupt Handling

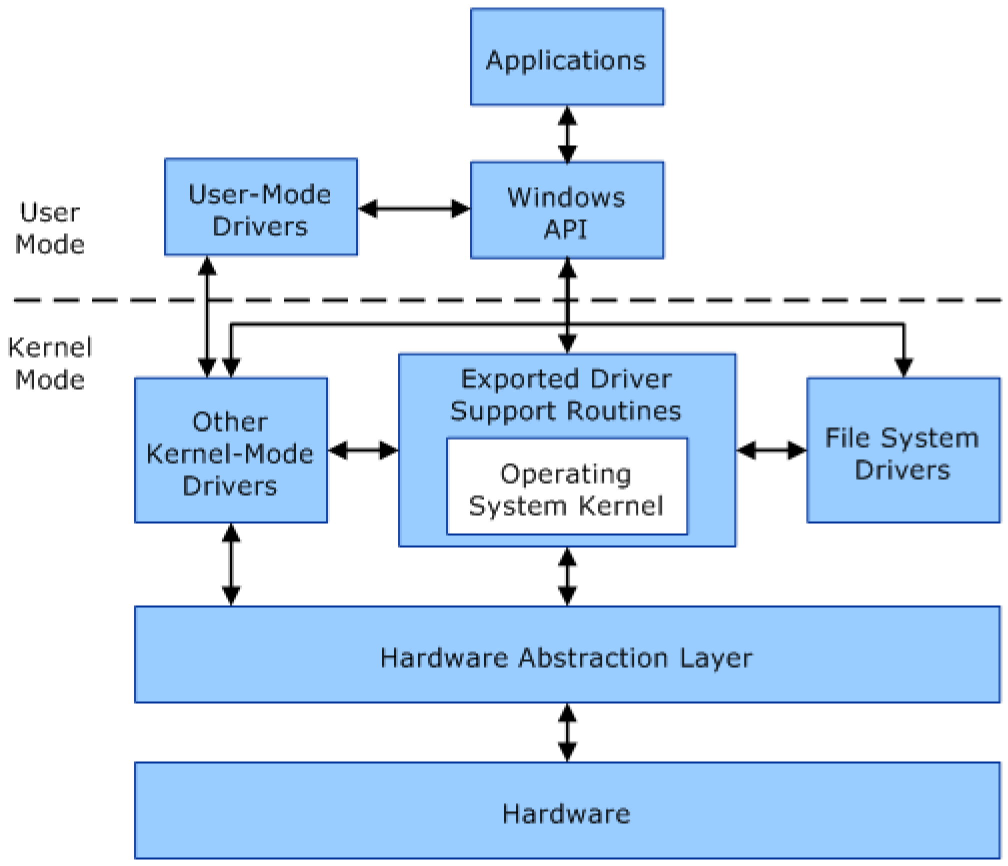
While executing the processes, there are conditions where tasks with more priority need to be handled first. In these cases, the kernel has to interrupt in-between the execution of the current process and handle tasks with more priority which has arrived in between.

5) I/O Communication

As the Kernel manages all the devices connected to it, so it is also responsible for handling all sorts of input and output that is exchanged through these devices. So, all the information that the system receives from the user and all the output that the user is provided with via different applications is handled by the Kernel.

What is a Kernel in OS

Now that we know that its a core program in the OS, one should also know it is also the first program that loads after the bootloader. It then does all the talking between the hardware and the software or applications. So if you launch a program, the user interface sends a request to Kernel. The Kernel then sends a request to CPU, Memory to assign processing power, memory, and other things so the application can run smoothly in the front end.



You can imagine Kernel as a translator. It converts input/output requests from software into an instruction set for the CPU and GPU. In simple words, its a layer between the software and the hardware which makes everything possible. The kernel manages the following:

1. CPU/GPU
2. Memory
3. Input/Output or IO devices
4. Resource management
5. Memory management
6. Device management
7. System calls.

User processes can access kernel-space only through the use of system calls. If a program tries to access directly, it will result in a fault.

Kernel Security & Protection

The kernel also protects the hardware. If there is no protection, any program will be able to carry out any task on the computer, including crashing your computer, corrupting data, etc.

In modern-day computers, security is implemented on the hardware level. For example, Windows will not load drivers which are not from a trusted source and certified using signature. Secure Boot and Trusted Boot are classic examples.

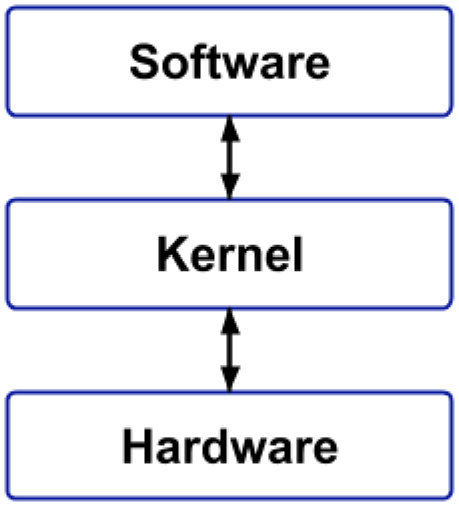
Secure Boot: It is a security standard developed by members of the PC industry. It helps you protect your system from malicious programs by not allowing any unauthorized applications to run during the system start-up process. The feature makes sure that your PC boots using only software that is trusted by the PC manufacturer. So, whenever your PC starts, the firmware checks the signature of each piece of boot software, including firmware drivers (Option ROMs) and the operating system. If the signatures are verified, the PC boots and the firmware gives control to the operating system.
Trusted Boot: It uses the Virtual Trusted Platform Module (VTPM) to verify the digital signature of the Windows 10 kernel before loading it. In turn, it confirms every other component of the Windows startup process, including the boot drivers, startup files, and ELAM. If a file has been altered or changed to any extent, the bootloader detects it and refuses to load it by recognizing it as the corrupted component. In short, it provides a chain of trust for all the elements during boot.

Types of Kernel

In general, there are five types of Kernel. They are:

1. Monolithic Kernels

Monolithic Kernels are those Kernels where the user services and the kernel services are implemented in the same memory space i.e. different memory for user services and kernel services are not used in this case. By doing so, the size of the Kernel is increased and this, in turn, increases the size of the Operating System. As there is no separate User Space and Kernel Space, so the execution of the process will be faster in Monolithic Kernels.



Advantages:

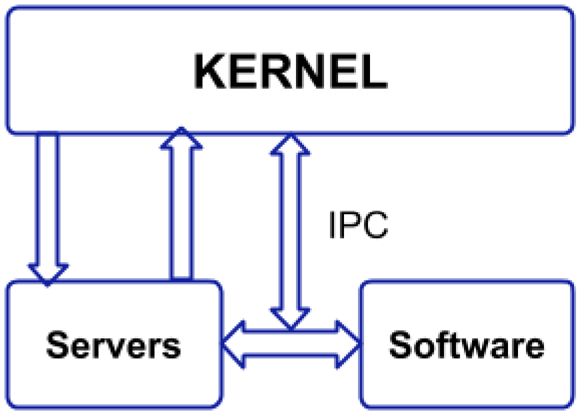
- It provides CPU scheduling, memory scheduling, file management through System calls only.
- Execution of the process is fast because there is no separate memory space for user and kernel.

Disadvantages:

- If any service fails, then it leads to system failure.
- If new services are to be added then the entire Operating System needs to be modified.

2. Microkernel

A Microkernel is different from Monolithic kernel because in a Microkernel, the user services and kernel services are implemented into different spaces i.e. we use User Space and Kernel Space in case of Microkernels. As we are using User Space and Kernel Space separately, so it reduces the size of the Kernel and this, in turn, reduces the size of Operating System.



As we are using different spaces for user services and kernel service, so the communication between application and services is done with the help of message parsing and this, in turn, reduces the speed of execution.

Advantages:

- If new services are to be added then it can be easily added.

Disadvantages:

- Since we are using User Space and Kernel Space separately, so the communication between these can reduce the overall execution time.

3. Hybrid Kernel

A Hybrid Kernel is a combination of both Monolithic Kernel and Microkernel. It makes the use of the speed of Monolithic Kernel and the modularity of Microkernel.

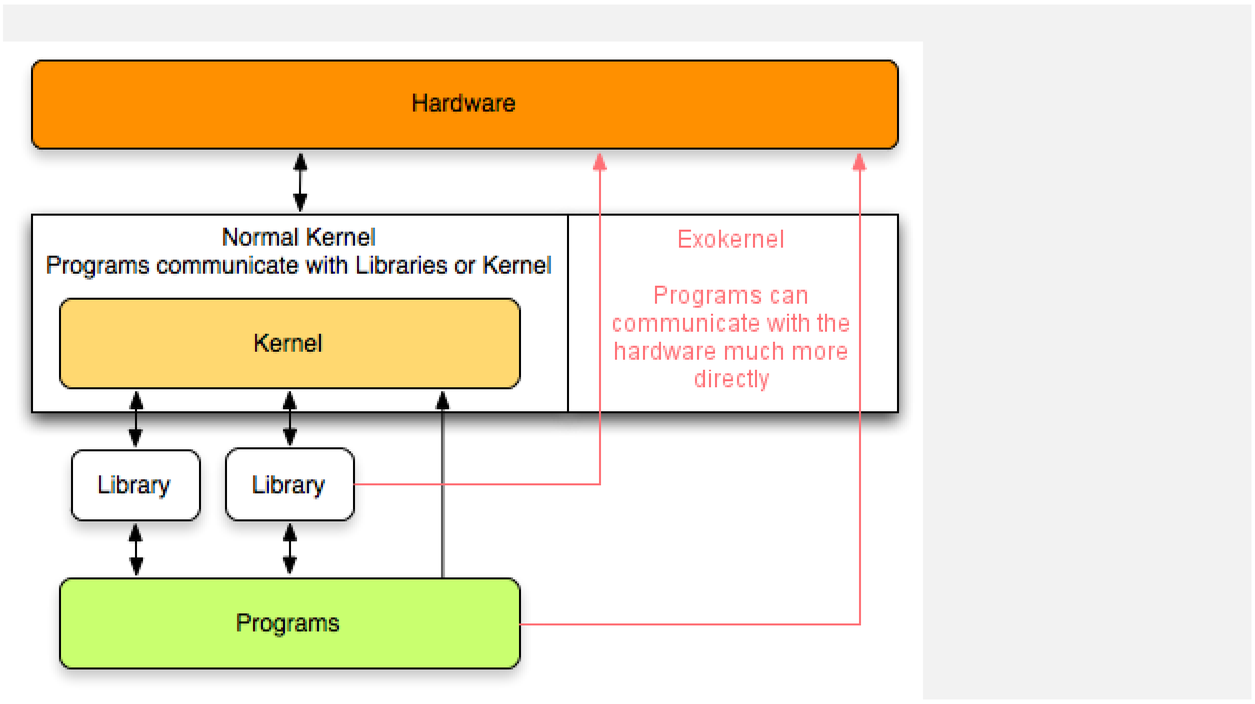
Hybrid kernels are micro kernels that have some "non-essential" code in kernel-space in order for the code to run more quickly than it would be in user-space. So, some services such as network stack or filesystem are run in Kernel space to reduce the performance overhead, but still, it runs kernel code as servers in the user-space.

4. Nanokernel

In a Nanokrnel, as the name suggests, the whole code of the kernel is very small i.e. the code executing in the privileged mode of the hardware is very small. The term nanokernel is used to describe a kernel that supports a nanosecond clock resolution.

5.Exokernel

Exokernel is an operating system developed at the MIT that provides application-level management of hardware resources. This architecture is designed to separate resource protection from management to facilitate application-specific customization.
Let's try to understand how this actually works.


Hardware
Normal Kernel
Programs communicate with Libraries or Kernel
Kernel
Exokernel
Programs can communicate with the hardware much more directly
Library
Library
Programs

The ultimate idea behind the development of exokernel is to impose as few abstractions as possible on the developers of the applications by providing them with the freedom to use the abstractions as and when needed. This ensures that there is no forced abstraction, which is what makes exokernel different from micro-kernels and monolithic kernels. But, how does exokernel support this?

This is done by moving all the hardware abstractions into untrusted user-space libraries called “library operating systems” (libOS), which are linked to applications call the operating system on their behalf. So basically, the kernel allocates the basic physical resources of the machine (disk blocks, memory, and processor time) to multiple application programs, and each program decides on what to do with these resources.

For an example, an application can manage its own disk-block cache, it can also share the pages with the other applications, but the exokernel allows cached pages to be shared securely across all applications. Thus, the exokernel protects pages and disk blocks, but applications manage them.

Of course, not all applications need customized resource management. At these instances, the applications can be linked with the support libraries that implement the abstractions that the applications need. However, library implementations are unprivileged and can therefore be modified or replaced at the user’s needs as well. This helps the programmers to choose what level of abstraction they want, high, or low.

Principles of Exokernels

1. Separate protection and management : Resource management is restricted to functions necessary for protection.
2. Expose allocation : Applications allocate resources explicitly.

3. Expose name : Exokernels use physical names wherever possible.

4. Expose revocation : Exokernels let applications to choose which instance of a resource to give up.

5. Expose information : Exokernels expose all system information and collect data that applications cannot easily derive locally.

Advantages of Exokernels

1. Significant performance increase.
2. Applications can make more efficient and intelligent use of hardware resources by being aware of resource availability, revocation and allocation.
3. Ease development and testing of new operating system ideas. (New scheduling techniques, memory management methods, etc)

Disadvantages of Exokernels

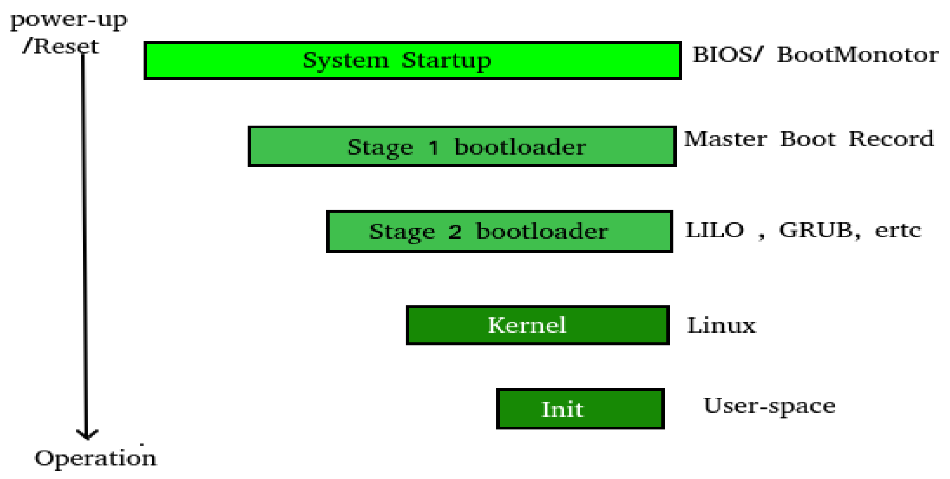
1. Complexity in design of exokernel interfaces.

2. Less consistency.

What happens when we turn on computer?

A computer without a program running is just an inert hunk of electronics. The first thing a computer has to do when it is turned on is to start up a special program called an operating system. The operating system's job is to help other computer programs to work by handling the messy details of controlling the computer's hardware.

An overview of the boot process



The boot process is something that happens every time you turn your computer on. You don't really see it, because it happens so fast. You press the power button come back a few minutes later and Windows XP, or Windows Vista, or whatever Operating System you use is all loaded.

The BIOS chip tells it to look in a fixed place, usually on the lowest-numbered hard disk (the boot disk) for a special program called a boot loader (under Linux the boot loader is called Grub or LILO). The boot loader is pulled into memory and started. The boot loader's job is to start the real operating system.

Functions of BIOS

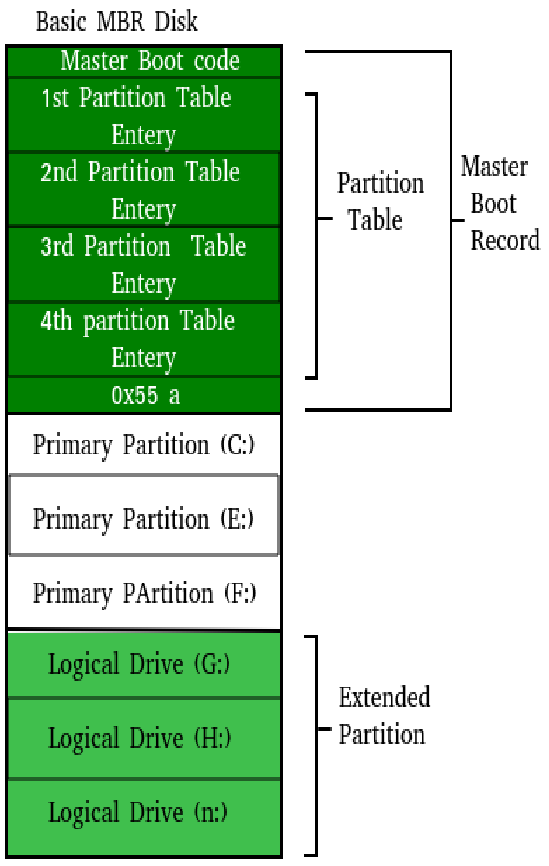
POST (Power On Self Test) The Power On Self Test happens each time you turn your computer on. It sounds complicated and that's because it kind of is. Your computer does so much when its turned on and this is just part of that.

It initializes the various hardware devices. It is an important process so as to ensure that all the devices operate smoothly without any conflicts. BIOSes following ACPI create tables describing the devices in the computer.

The POST first checks the bios and then tests the CMOS RAM. If there is no problem with this then POST continues to check the CPU, hardware devices such as the Video Card, the secondary storage devices such as the Hard Drive, Floppy Drives, Zip Drive or CD/DVD Drives. If some errors found then an error message is displayed on the screen or a number of beeps are heard. These beeps are known as POST beep codes.

Master Boot Record

The Master Boot Record (MBR) is a small program that starts when the computer is booting, in order to find the operating system (eg. Windows XP). This complicated process (called the Boot Process) starts with the POST (Power On Self Test) and ends when the Bios searches for the MBR on the Hard Drive, which is generally located in the first sector, first head, first cylinder (cylinder 0, head 0, sector 1).
A typical structure looks like:



The bootstrap loader is stored in computer's EPROM, ROM, or another non-volatile memory. When the computer is turned on or restarted, it first performs the power-on-self-test, also known as POST. If the POST is successful and no issues are found, the bootstrap loader will load the operating system for the computer into memory. The computer will then be able to quickly access, load, and run the operating system.

init

init is the last step of the kernel boot sequence. It looks for the file */etc/inittab* to see if there is an entry for *initdefault*. It is used to determine initial run-level of the system. A run-level is used to decide the initial state of the operating system.
Some of the run levels are:
Level

0 –> System Halt

1 –> Single user mode

3 –> Full multiuser mode with network

5 –> Full multiuser mode with network and X display manager

6 –> Reboot

The above design of init is called SysV- pronounced as System five. Several other implementations of init have been written now. Some of the popular implementatios are systemd and upstart. Upstart is being used by ubuntu since 2006. More details of the upstart can be found here.

The next step of init is to start up various daemons that support networking and other services. X server daemon is one of the most important daemon. It manages display, keyboard, and mouse. When X server daemon is started you see a Graphical Interface and a login screen is displayed.