Introduction to Operating Systems

In this lecture, we have learned about:

Detailed Definition of Operating Systems

OS is a piece of code that -

- Has privileged access to underlying hardware
- Hides complexity of the hardware
- Manages the resources

Makes sure that the apps are isolated and protected from one another.

Different analogies and concepts related to the OS:

- o Hardware resources - kitchen staff
- o Worker/ waiters - Applications
- o Customers of the restaurant - Users of computer system
- o the manager was shown to manage the resources of the kitchen and there were 2 functions which were highlighted: Acting as a resource manager to make efficient use of them and giving a seamless experience to waiters and customers
- o The operating system is analogous to Manager and the two functions were: ▪ Resource Manager

Hides complexity of dealing with hardware from users

Abstraction

Operating system is analogous to Manager and one of the functions that a manager performs is that he/she hides the complexity of dealing of chefs with customers directly, similarly in a computer OS hides the complexity of its hardware from users and this is known as Abstraction.

Example of abstraction: when you play a song through the Music player in the system, it doesn't know which sound card is there in the system, it just knows its function which is to play music.

Arbitration

Another function that a manager performs is that he/she is responsible for the allocation of different resources and tasks properly in the kitchen. Similarly, Arbitration means that the operating system provides the resources needed for application software to run. When a computer system is running, a lot of programs are running in the backend, each of these programs needs CPU time, memory, access to the disk, and several other system resources, it is the responsibility of the operating system to provide these resources to the program that needs them in an orderly fashion and does not create any conflict.

Example of Arbitration: When we copy the files from the USB disk to the internal hard disk, it is the responsibility of the operating system to provide the resource like the CPU time, memory to store the file when we click on copy from the USB disk and access to the internal hard disk when we click on paste at the desired location in the internal disk. In this case, OS arbitrates the CPU, memory, USB disk, and internal disk.

Components of OS

The components of OS can be best summarised using image:

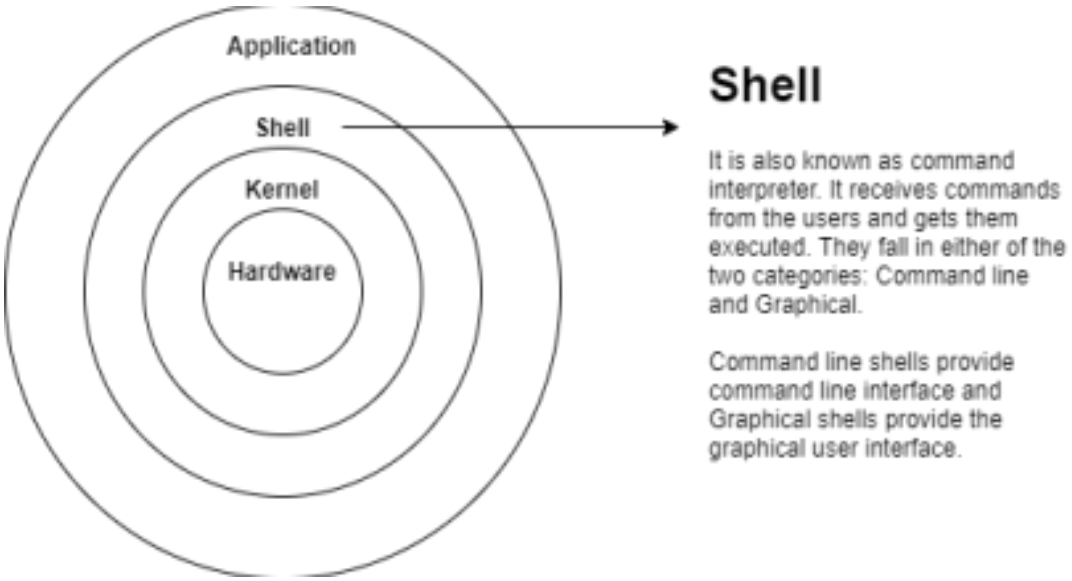


Figure 1

Kernel

A Kernel is the central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is the core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Types of Kernels

Monolithic Kernel

Example: Unix, Linux, etc.

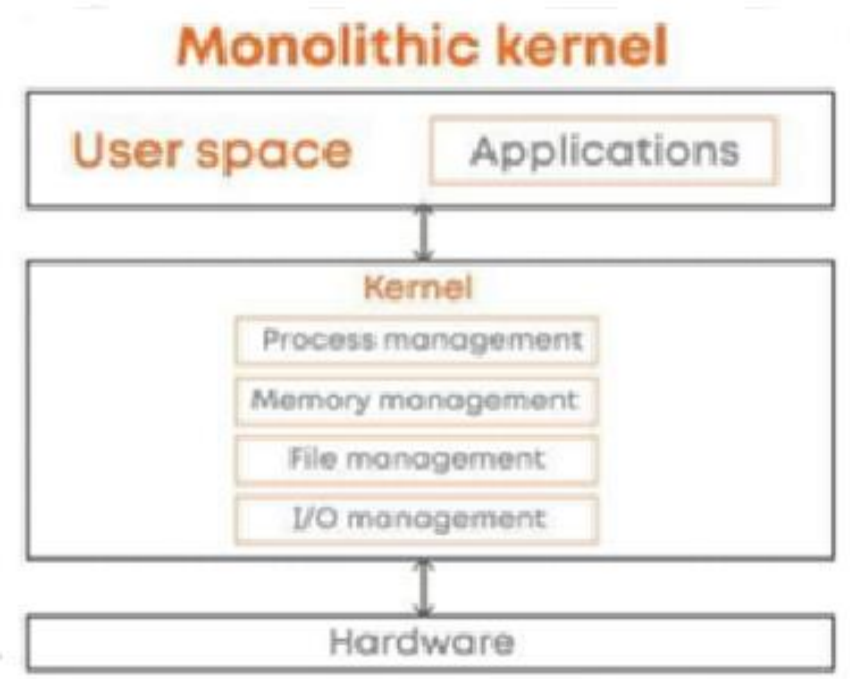


Figure 2

Micro Kernel

Examples: L4Linux, Minix, etc.

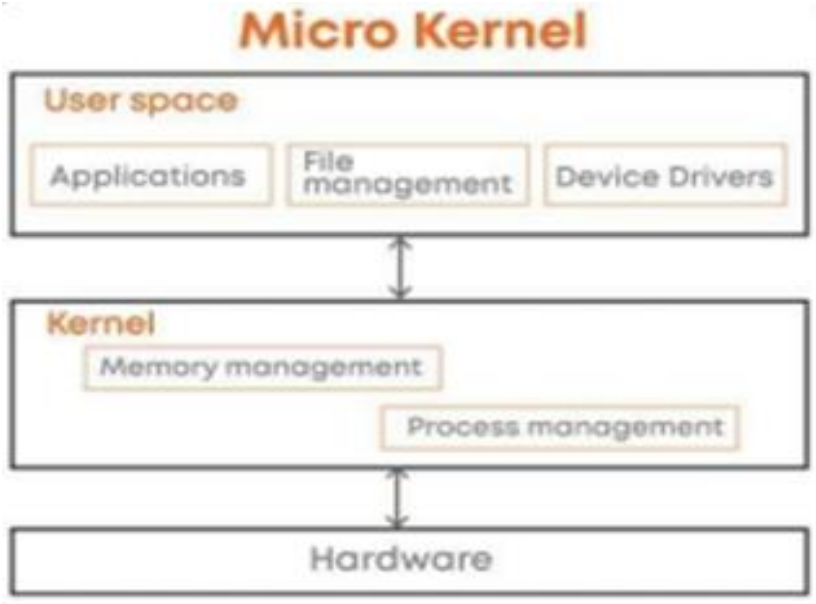


Figure 3

Hybrid Kernel

Examples: Windows 7, Windows 10, Mac OS, etc.

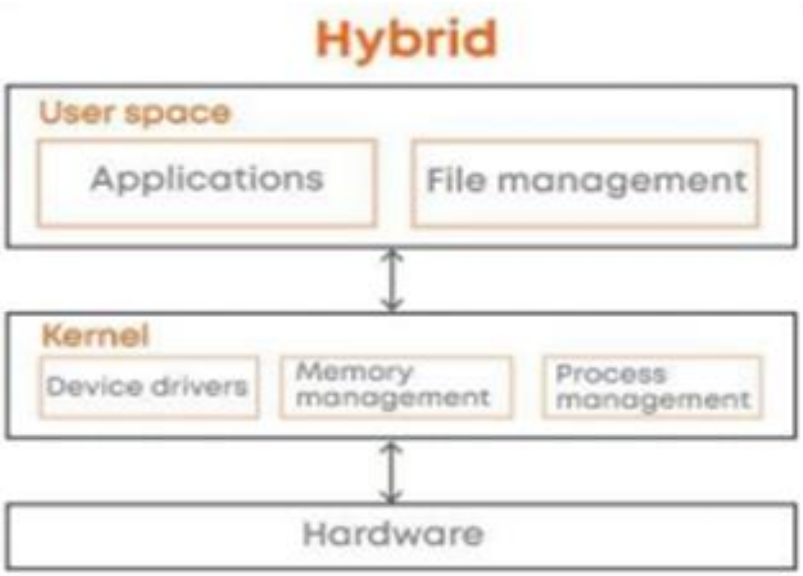


Figure 4

Shell

A **shell**, also known as a command interpreter, is that part of the operating system that receives commands from the users and gets them executed.

System calls

A system call is a mechanism using which a user program can request a service from the kernel for which it does not have permission to perform. User programs typically do not have permission to perform operations like accessing I/O devices and
communicating with other programs.

A user program invokes system calls when it requires such services.

System calls provide an interface between a program and the operating system.
System calls are of different types.

Example – fork, exec, getpid, getppid, wait, exit.

Operation mode

1. User Mode

When the computer system runs user applications like creating a text document or using any application program, then the system is in user mode. When the user application requests for a service from the operating system or an interrupt occurs or system call, then there will be a transition from user to kernel mode to fulfil the requests.
To switch from kernel mode to user mode, the mode bit should be 1.

2. Kernel Mode

When the system boots, hardware starts in kernel mode and when the operating system is loaded, it starts the user application in user mode. To provide protection to the hardware, we have privileged instructions which execute only in kernel mode. If a user attempts to run privileged instruction in user mode, then it will treat the instruction as illegal and trap the OS.
To switch from user mode to kernel mode bit should be 0.

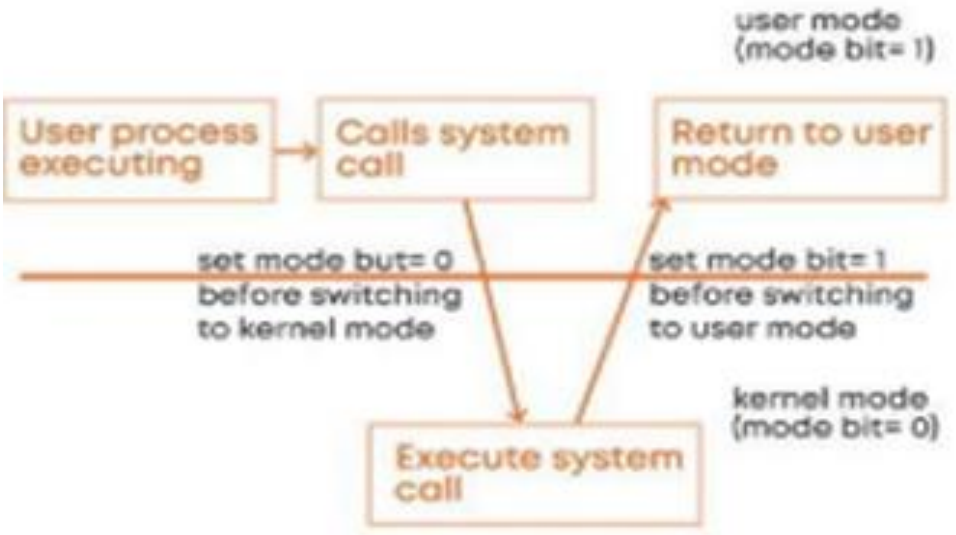


Figure 5

Note: In the above diagram, the mode bit is spelled wrong. The line should be: “set mode bit=0 before switching to kernel mode”.

Main Elements of OS Design Principles:

While designing the OS, three main elements are considered:

1. **Abstraction**:
   The method that Operating systems use to hide the complexity of the underlying hardware.
2. **Mechanism**:
   To fulfil the abstraction i.e., hide the complexities; OS follows different methods using different system calls. Different system calls include
   a. fork() to create a new process.
   b. create() to create a new file.
   c. open() to open a file or a device.
   d. allocate() to associate memory
   e. exit() to terminate a process.
   f. exec() to execute a process….and many more.

So, system calls are basically a mechanism that OS uses to achieve abstraction

3. **Policies** are ways or plan to make a decision. E.g.: there can be a policy that limits the maximum number of sockets that an application can have access to. Another example can be that there can be a policy that all allows memory allocation based on a first come first serve basis.

Most operating systems implement the Least Recently Used or LRU policy for allocating memory. This policy defines that the least used memory page will be replaced by the new memory page.

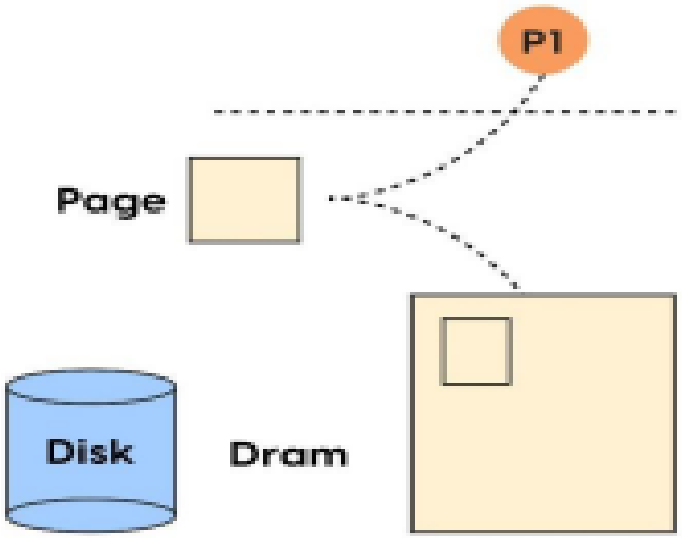


Figure 6

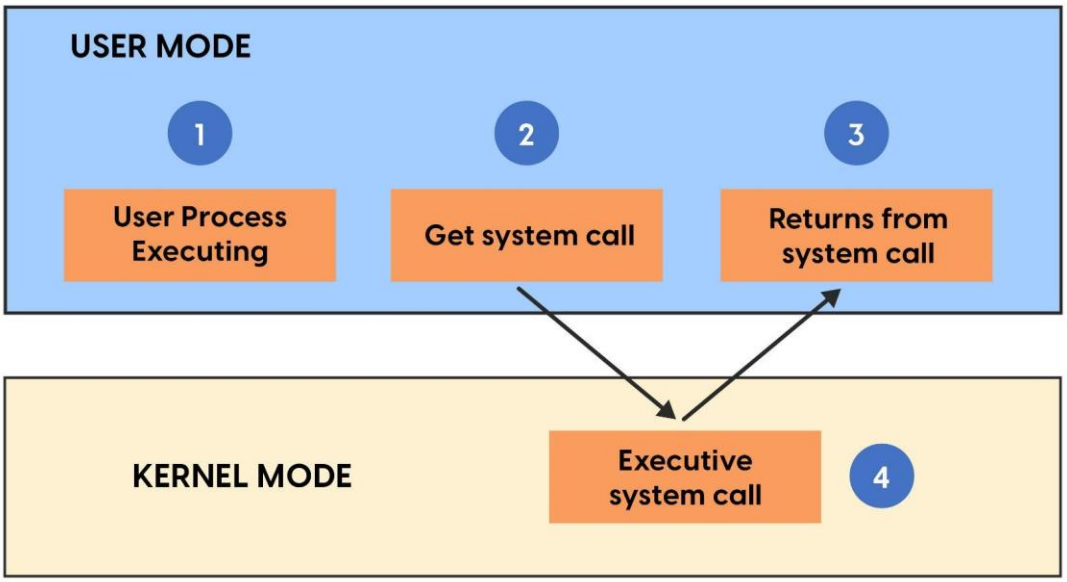
In the above example, Abstraction is the memory page. A mechanism is a method that is used to grant memory pages to the process. It may use mmap(), a system call that we use for allocating memory. The policy is the LRU.

OS design principles:

1. **Separation of mechanism and policies**: mechanism defines what method needs to be followed to do something and policies define the rules that should be taken into account when following a mechanism. So, there should be a clear distinction between the mechanism and policy and a mechanism should support many policies. It should allow the different applications to use the same mechanism and allow implementation with different policies. Also, separating mechanisms and policies allows experimenting with newer policies without breaking the existing mechanism.
2. **Consider common use cases**: When designing the OS, questions like the OS will be for what kind of users, what will be the workload running on the OS, what kind of application will be running on these OS, etc. Based on the answers, the required policies, mechanisms, and abstraction can be implemented. E.g., we use desktop OS like Windows 7 or Windows 10 on our personal computer, Android or IOS on mobile devices, and Server OS like RedHat or Windows

Server 2016 on the server. Each of these OSs fulfils a different set of use cases.

3. **The OS should macro-manage and not micro-manage**. It means that the OS should come in the picture only when required. It should make each and every decision on its own. If it does that, then the OS operation will be very time-consuming and memory extensive. We have seen the example of the OS macro-management technique when we were discussing system calls. When we run a program, OS only comes into the picture when the program requests resources from the OS i.e. from the kernel. That time the control switches from user-mode to kernel-mode. Once the resource is granted to the process, the control goes to the process and OS does not interfere in its execution.



Architechure of the System Call

Figure 7

The above image provides the information about the architecture of the system call. This image shows the relationship between User mode and kernel mode. This shows us how Design Principles are implemented in real life.