OPERATING SYSTEM

What is a Kernel?

A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory etc). The main tasks of the kernel are:

- Process management
- Device management
- Memory management
- Interrupt handling
- I/O communication
- File system...etc.

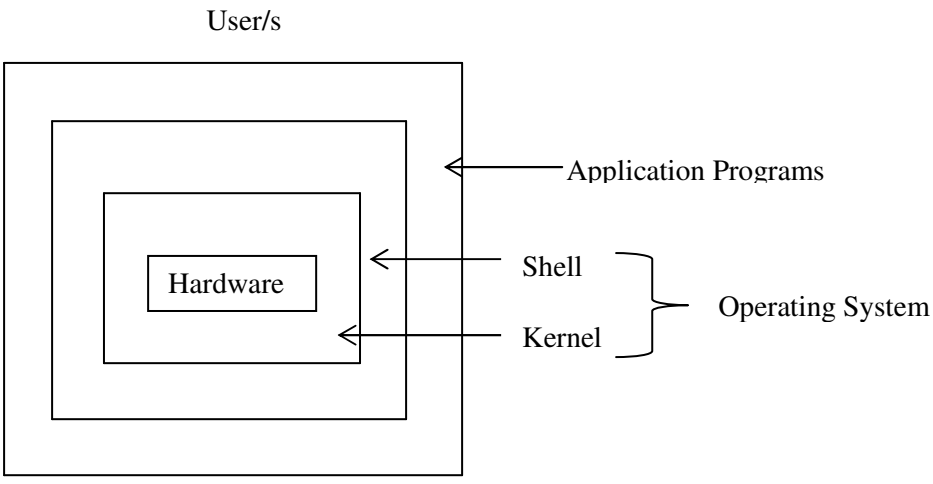
What is the relationship between Shell and Kernel?

The **shell** acts as an interface between the user and the **kernel**. When a user logs in, the login program checks the username and password, and then starts another program called the **shell**. The **shell** is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out.

- ***What happens if we have ONLY the kernel BUT NO shell?*** You then have a machine with the actual OS but there is NO way you can use it. There is no "interface" for the human to interact with the OS and hence the machine. (Assuming GUIs don't exist, for simplicity)
- ***What happens if we have ONLY the shell BUT NO kernel?***

This is impossible. Shell is a program provided by the OS so that you can interact with it. Without the kernel/OS nothing can execute (in a sense, not 100% true though, but you get the idea).

A shell is just a program that offers some functionality that runs on the OS. The kernel is the "essence/core" of the OS.



Prof. Srijoni Maitra, Bethune College

What are the types of Kernels?

Kernels may be classified mainly in two categories –

1. Monolithic
2. Micro Kernel

1 Monolithic Kernels

Earlier in this type of kernel architecture, all the basic system services like process and memory management, interrupt handling etc. were packaged into a single module in kernel space. This type of architecture led to some serious drawbacks like 1) Size of kernel, which was huge. 2) Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours.

In a modern day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. With this approach, maintainability of kernel became very easy as only the concerned module needs to be loaded and unloaded every time there is a change or bug fix in a particular module. So, there is no need to bring down and recompile the whole kernel for a smallest bit of change.
Linux follows the monolithic modular approach

2. Microkernels

This architecture majorly caters to the problem of ever growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file system etc. to run in user space. This reduces the kernel code size and also increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional.

In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC). e.g., we have servers for device drivers, network protocol stacks, file systems, graphics, etc. Microkernel servers are essentially daemon programs like any others, except that the kernel grants some of them privileges to interact with parts of physical memory that are otherwise off limits to most programs. This allows some servers, particularly device drivers, to interact directly with hardware. These servers are started at the system start-up. So, what the bare minimum that microkernel architecture recommends in kernel space?

- Managing memory protection
- Process scheduling
- Inter Process communication (IPC)

Apart from the above, all other basic services can be made part of user space and can be run in the form of servers.

QNX follows the Microkernel approach