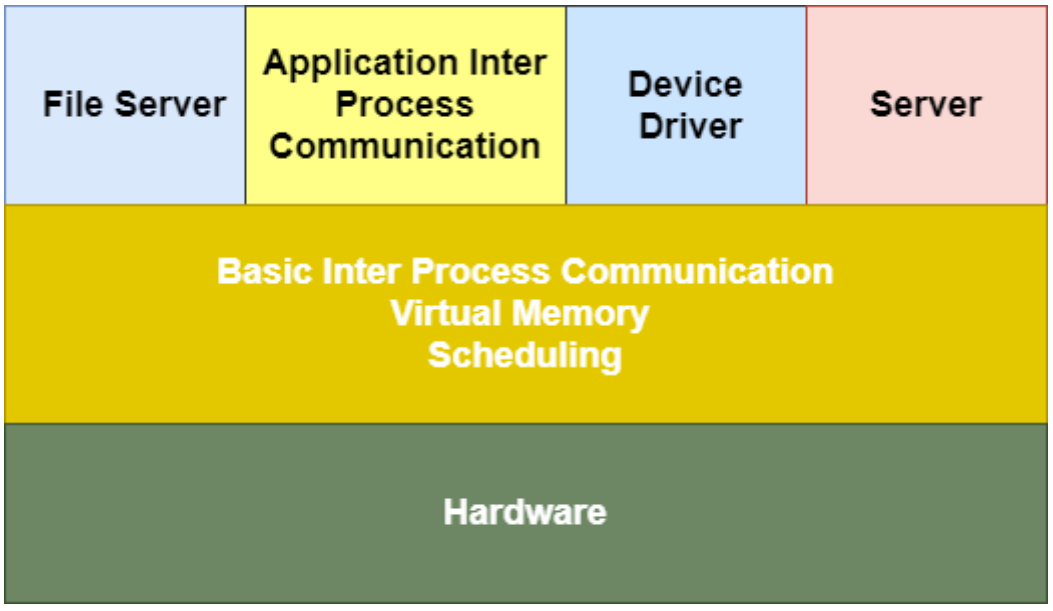
Kernel

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.

Kernel Architecture



Micro Kernel Architecture

Types of Kernel

1. Monolithic Kernel: Here, the OS and Kernel both run in the same memory space and are suitable where security is not a significant concern. It results in faster access, but if there is a bug in the device driver, the entire system crashes.

Example: Unix, Linux, Open VMS, XTS-400 etc.

2. Microkernel: It's a stripped-down version of Monolithic Kernel where the Kernel itself can do most of the job, and there is no need for an extra GUI..A microkernel is much smaller in size than a conventional kernel and supports only the core operating system functionalities.

Example: Mach, L4, AmigaOS, Minix, K42 etc.

3. Hybrid Kernel: This Kernel is what we see most. Windows, Apple's macOS. They are a mix of Monolithic Kernel and Microkernel. It moves out drivers but keeps system services inside the Kernel – similar to how drivers are loaded when Windows Starts the bootup process.

Example: Windows NT, Netware, BeOS etc.

4. Nano Kernel: If you need to have a kernel, but its majority of function is set up outside, then this comes into the picture.

Example: EROS etc.

5. Exo Kernel: This kernel only offers process protection and resource handling. However it is mostly used when you are testing out an inhouse project, and you upgrade to a better Kernel type.

Example: Nemesis, ExOS etc.

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 the 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 the 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 fulfill the requests.
To switch from kernel mode to user mode, 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 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 traps to OS.
To switch from user mode to kernel mode mode bit should be 0.