Operating system structure cont…

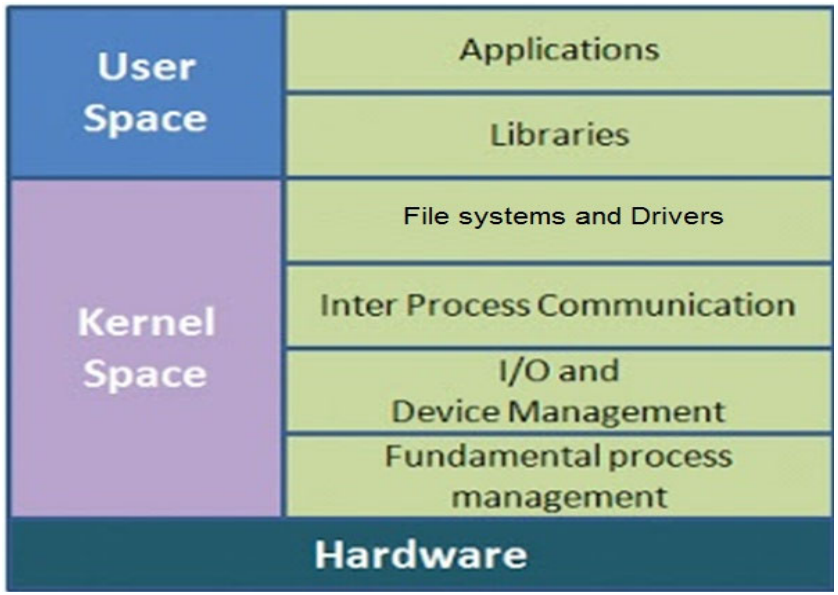
- Based on the separation of mechanism and policy, there are different types of architecture for operating systems which lead to different kernel designs:
  - Monolithic/Simple systems
  - Layered systems
  - Microkernel/ Client-server
  - Modular kernel
  - Hybrid

Monolithic/Simple kernel (یکپارچه)

- The first part "Mono" means "one". The second part "lithic" means "stony; it is like stone".

- A monolithic kernel is an OS architecture where the entire OS is working in the kernel space. Monolithic is one single program that contains all of the kernel code necessary to perform every kernel related task.

- Example of designs are **Contiki**, **TinyOS,** DOS and early Unix systems.



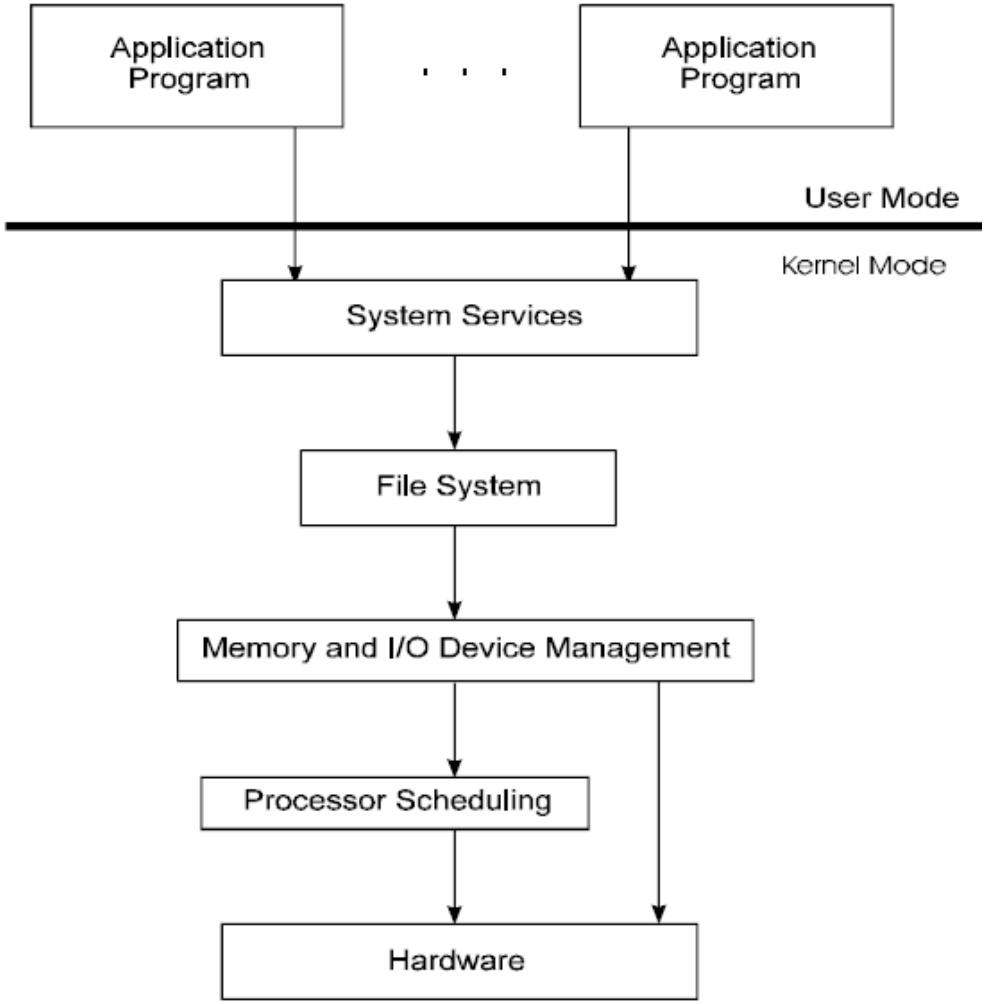
Monolithic kernel cont…

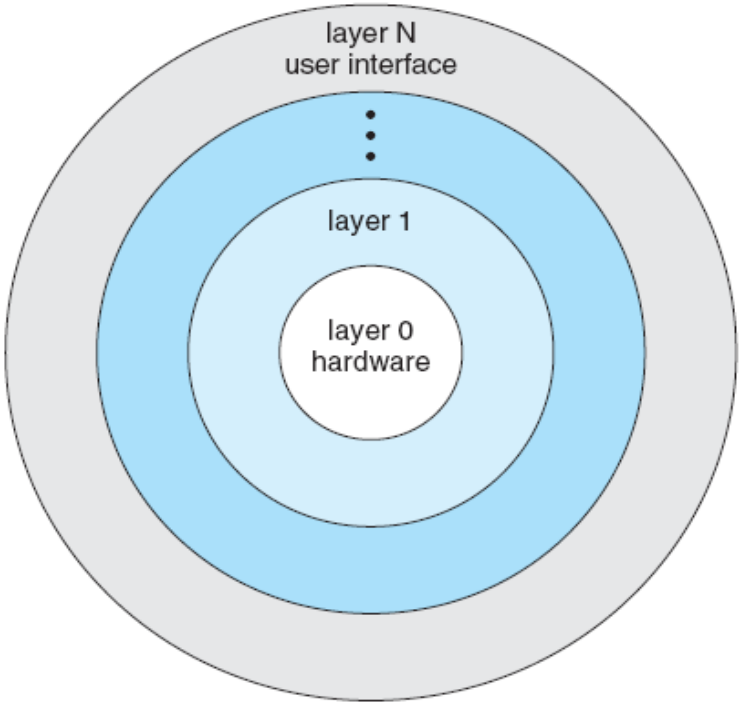
- Pros:
  - Little planning required; simple
- Cons:
  - Hard to extend and change
  - Eventually becomes extremely complex and poor performance. As monolithic OS expanded, the kernel became large and difficult to manage.
  - Unreliable, as a bug anywhere in the kernel can bring down the whole system

Layered OS

- A system can be made modular in many ways. One method is the layered approach, in which the OS is broken into some layers (levels) stacked them one on top of the other.
- The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.
- each layer provides a set of functions that other layer can call. The layers are selected so that each function at any particular level can only invoke/request services provided by lower layers.
- A layer does not need to know how these lower operations are implemented; it needs to know only what these operations do. Hence, each layer hides certain data structures, operations, and hardware from higher layers.

Layered OS cont…





Layered OS cont…

- Pros:
  - The main advantage of the layered approach is simplicity of construction and debugging. Rule is to **debug layer n before looking at layer n+1**. The first layer can be debugged without any concern for the rest of the system, because, by definition, it uses only the basic hardware (which is assumed correct) to implement its functions. Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on. If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged. Thus, the design and implementation of the system are simplified.

  - Layering makes it easier to enhance the OS; one entire layer can be replaced without affecting other parts of the system.

Layered OS cont…

- Cons:
  - What goes in what layers? For two functions X and Y, one must decide if X is above, at the same level, or below Y. Thus, the major difficulty with the layered approach involves appropriately defining the various layers.
  - Because a layer can use only lower-level layers, careful planning is necessary.

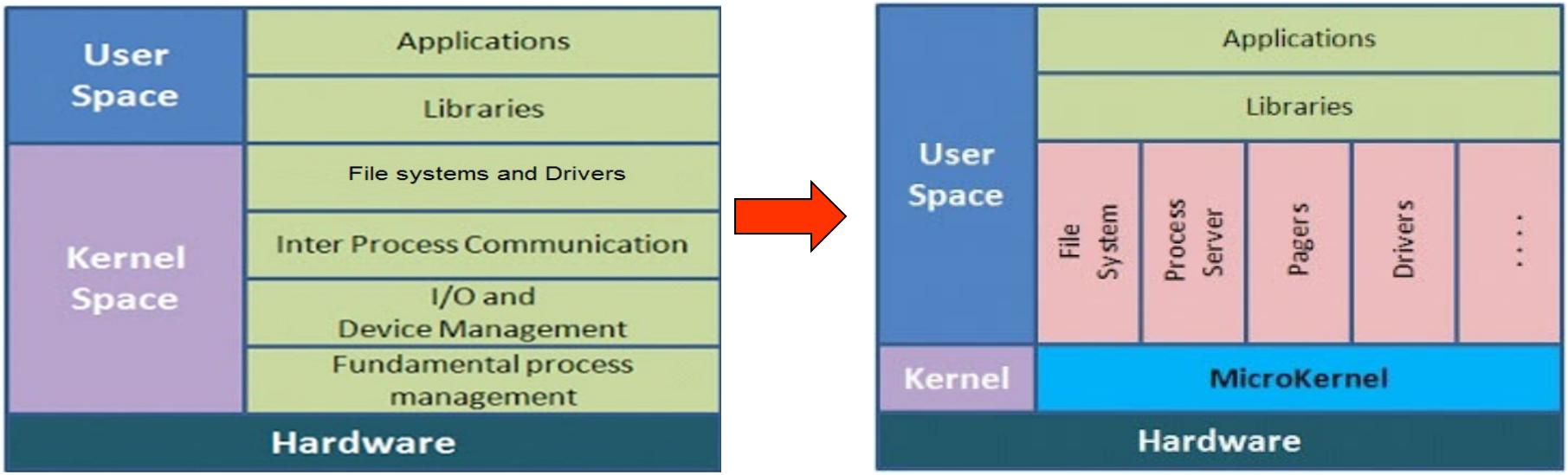
Layered OS cont…

- It can be less efficient because going through layers for each system call takes time, so there should be few layers. For instance, when a user program executes an I/O operation, it executes a system call that is trapped to the I/O layer, which calls the memory-management layer, which in turn calls the CPU-scheduling layer, which is then passed to the hardware. Each layer adds overhead to the system call; the result is a system call that takes longer than does one on a non-layered system.

5	User
4	User programs
3	I/O management
2	Operator - process communication
1	Memory management
0	CPU scheduling (CPU belongs to what program at the moment)

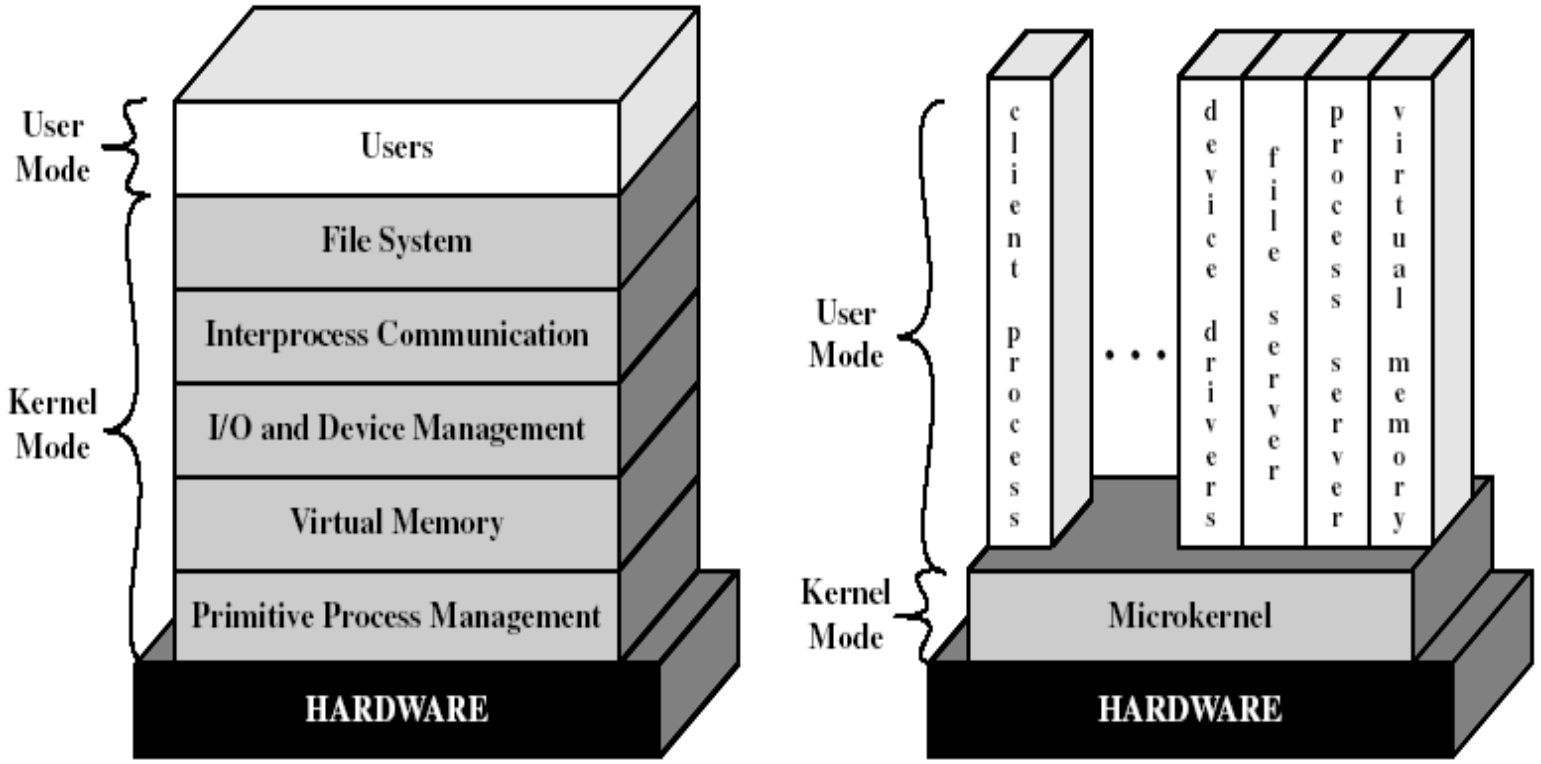
Microkernels/Client-server kernel approach

- We have already seen that as monolithic OS expanded, the kernel became large and difficult to manage.
- In order to deal with the disadvantages of monolithic kernels, the idea was that a kernel should be very small, by removing nonessential components as much as possible from the kernel space and putting it all in user space. The result is a smaller Kernel called a microkernel. This tiny OS kernel provides only basic primitive (process, memory, IPC).
- Example is: RIOT, FreeRTOS



Microkernels OS cont…

- The OS (kernel) is divided into several processes each of which implements a single set of services for example, I/O servers, memory server, process server. Each server runs in user mode, provides services to the requested client.



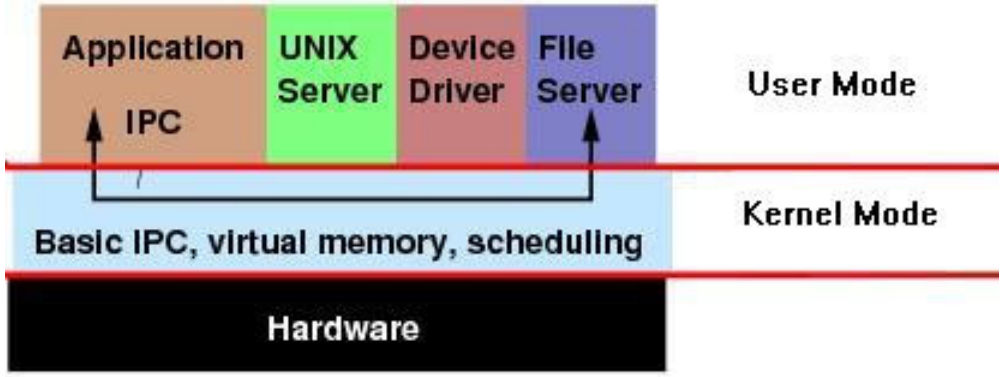
(a) Layered kernel

(b) Microkernel

Microkernels OS cont…

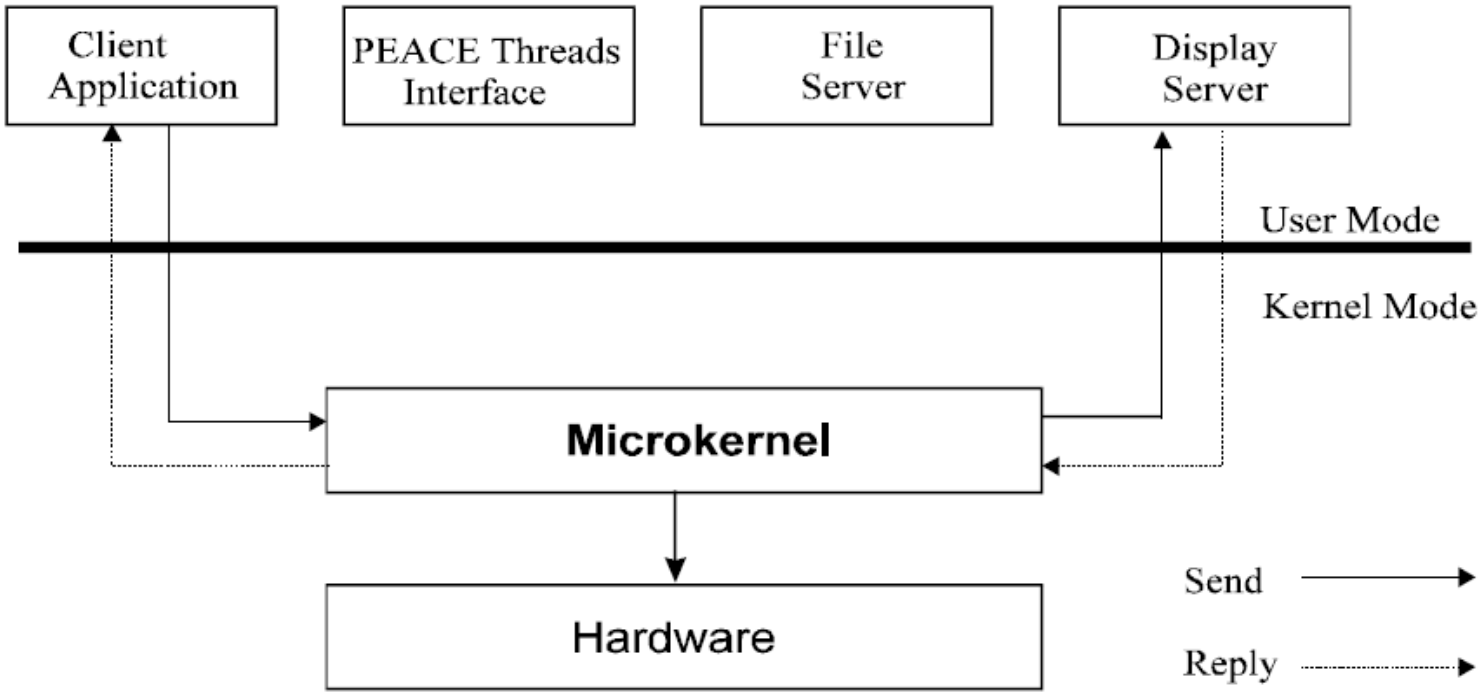
- All the kernel does is providing the communication between the clients and the servers:
  - A client is a user program.
  - A server is an external program that is granted special privileges by the kernel that normal programs do not have. These "privileges" may be direct access to hardware. Remember that microkernels are very minimal. They rely on external programs (servers) to help out.

- Servers needed by the kernel itself is normally loaded into memory before the kernel is executed.



Microkernels OS cont…

- The client program and server (service) never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel. The client, requests a service by sending a message. The kernel delivers the message to the appropriate server; the server then performs the operation and kernel delivers the results to the client in another message.



Microkernels OS cont…

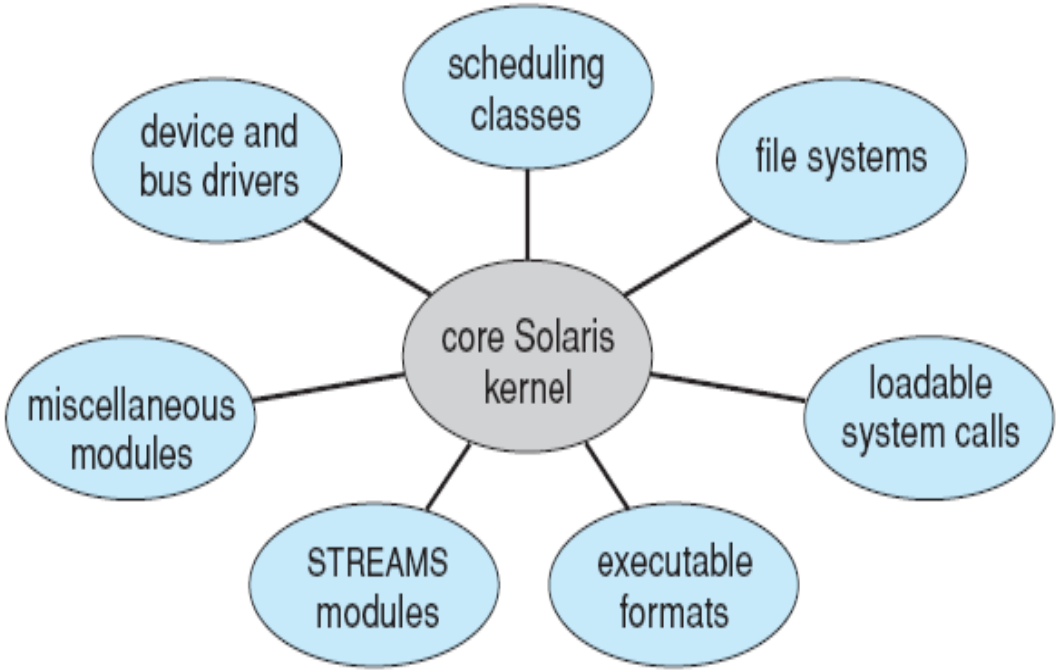
- Pros:
  - Extensibility: A minimal kernel results in easier maintenance and extending the OS because not so many processes are running in kernel mode. All new services are added to user space and consequently do not require modification of the kernel.
  - Reliability: The microkernel also provides more reliability, since most services are running as user processes. If a service fails, the rest of the operating system remains untouched.

Microkernels OS cont…

- Cons:
  - Microkernel can suffer from performance decreases due to increased system function overhead.
  - The proper division of functionality between the microkernel and its servers is the main issue.

Modular kernel approach

- This type of design is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X, as well as Windows. Here, the kernel has a set of core components linked in additional services via modules. Each core component is separate and talks to the others over known interfaces. Each core is loadable at boot time or at runtime as needed within the kernel.



Modular kernel vs. Layered kernel

- Similarity: modular kernel requires subsystems to interact with each other through carefully constructed interfaces

- Difference: modular kernel is more flexible than a layered system, because any module can call any other module. The layered kernel imposes a strict ordering of subsystems such that subsystems at the lower layers are not allowed to invoke operations corresponding to the upper-layer subsystems. There are no such restrictions in the modular-kernel approach, wherein modules are free to invoke each other without any constraints.

Modular kernel vs. Microkernel kernel

- Similarity: the primary module has only core functions and knowledge of how to load and communicate with other modules

- Difference: modular system is more efficient, because there is no message passing between modules in order to communicate. Therefore, we can say that module system has advantage of microkernels without the overhead problem.

Hybrid kernel

- Practically, most modern operating systems are not only one pure model. Instead, they try to take grab the best features of multiple design ideas resulting in hybrid systems that address performance, security, and usability issues.
  - Linux, Unix (FreeBSD), and Apple Mac OS X are monolithic, because having the operating system in a single address space provides very efficient performance. However, they are also modular, so that new functionality can be dynamically added to the kernel.
  - Windows Kernel is mostly monolithic plus microkernel.