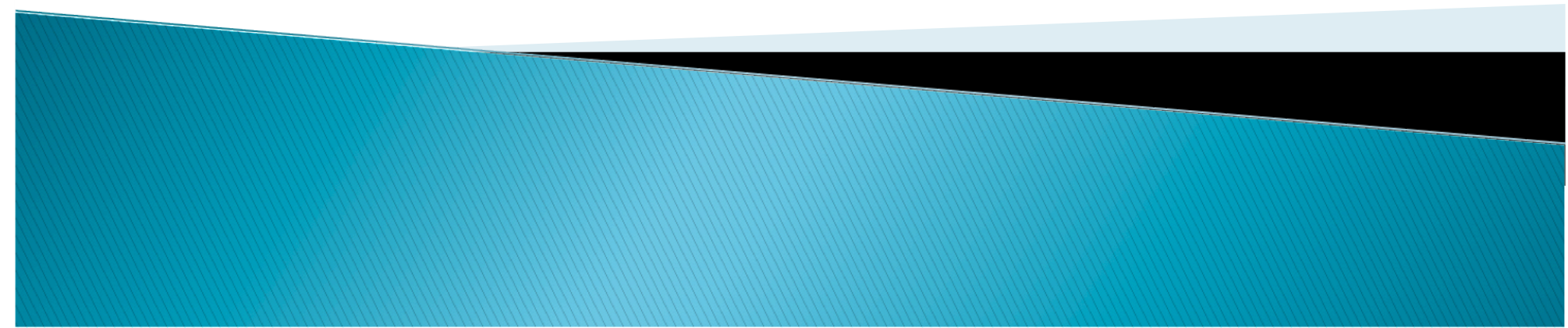
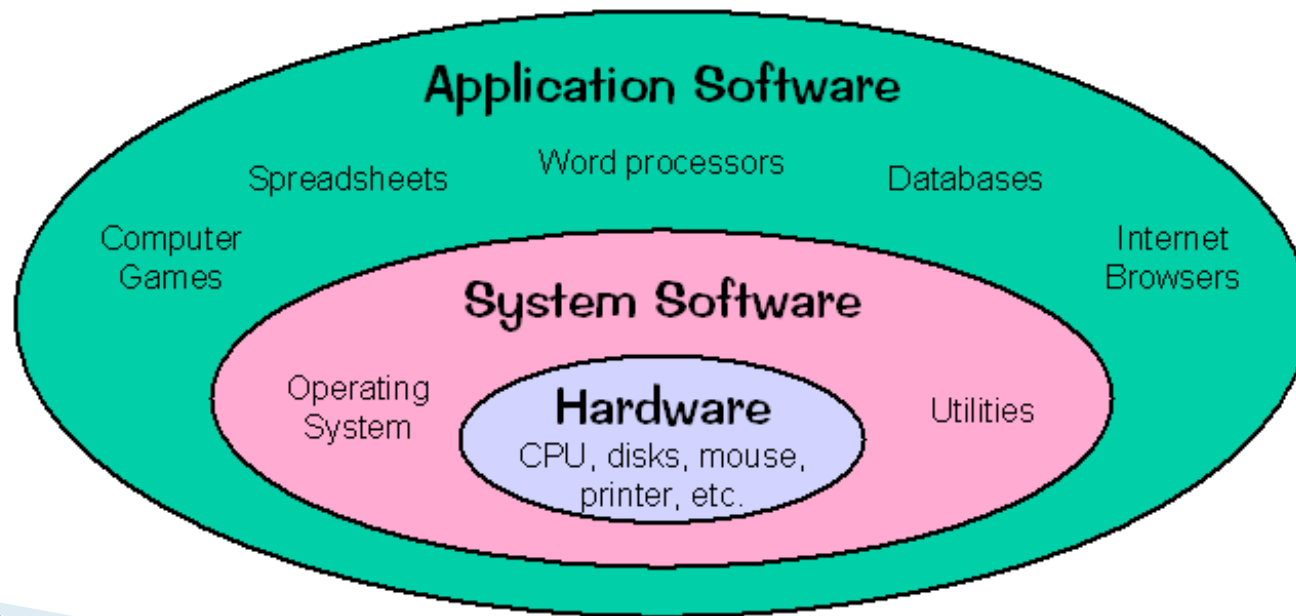


Chapter 2: Operating-System Structures



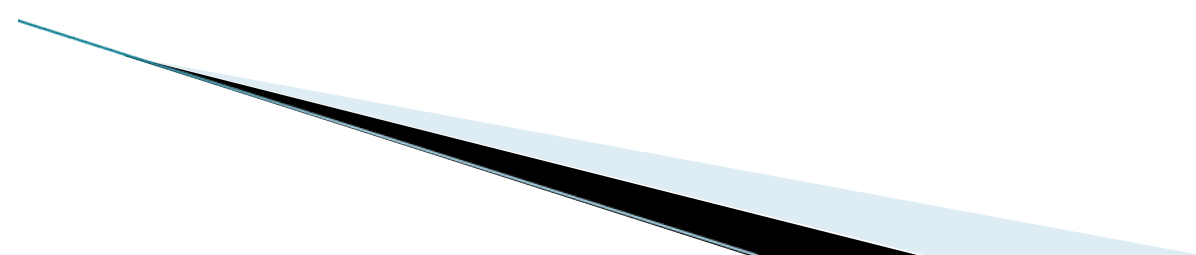
System Programs

- System software is a type of computer program that is designed to run a computer's hardware and application programs. It is the interface between the hardware and user applications. Operating system (OS) is the best-known example of system software. The OS manages all the other programs in a computer.



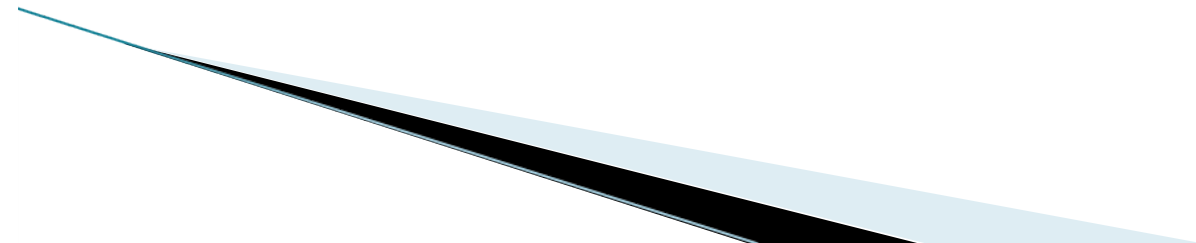
System Programs

- System programs provide a convenient environment for program development and execution.
- Some of them are simply user interfaces to system calls; others are considerably more complex.



Operating-System Structure

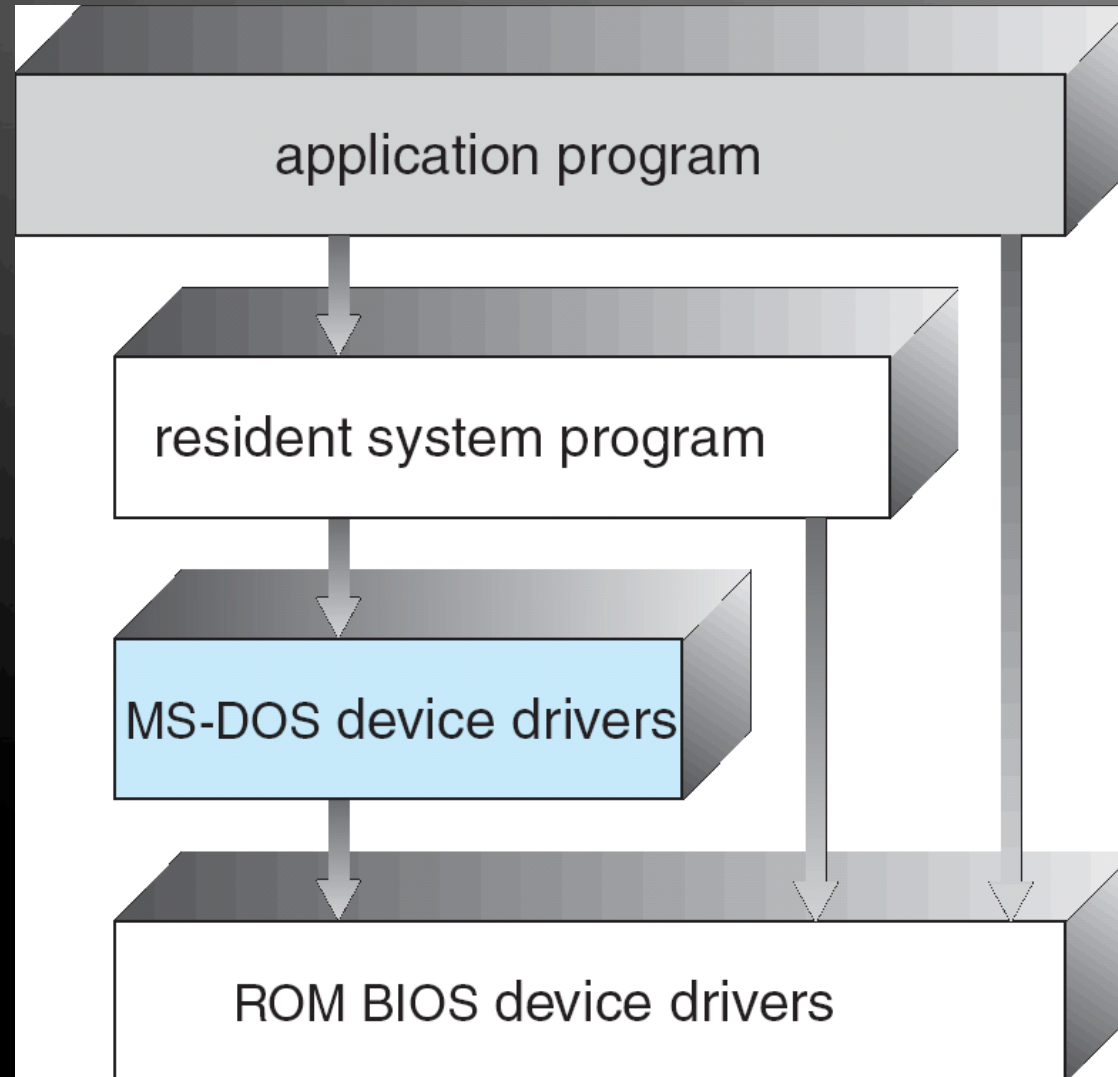
- A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily.
- Many approaches exist for the structuring of operating system. We will discuss each one by one.



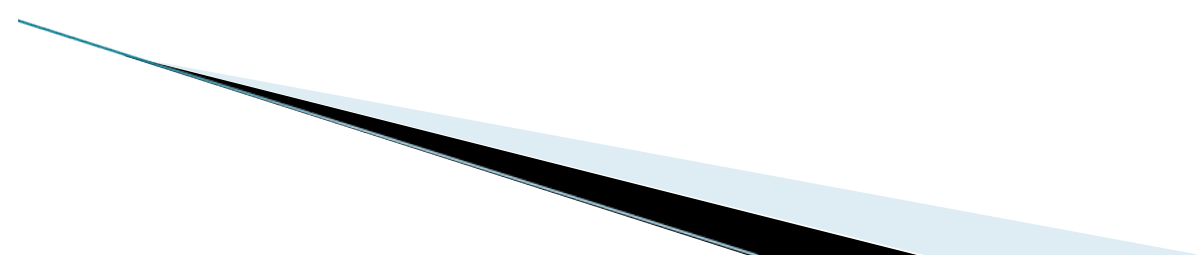
Simple Structure

- Many commercial systems do not have well-defined structures. They started as small, simple, and limited systems and then grew beyond their original scope.
- MS-DOS is an example of such a system. It was originally designed and implemented by a few people who had no idea that it would become so popular.
-
- It was written to provide the most functionality in the least space, so it was not divided into modules carefully.

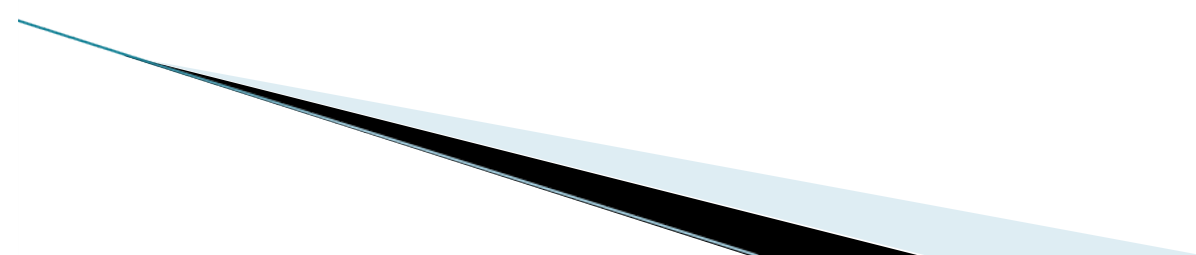
MS-DOS Layer Structure



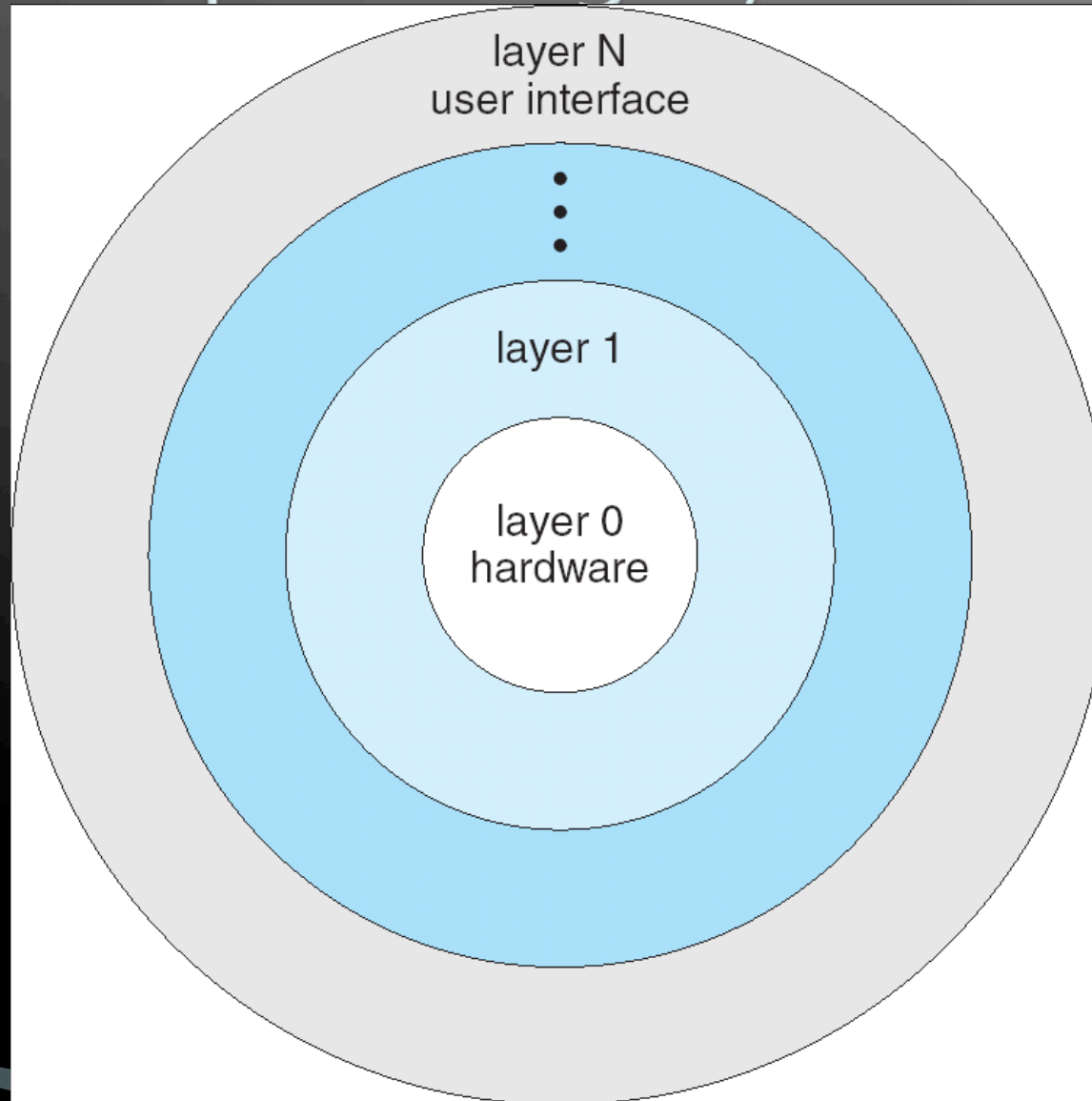
Simple Structure (cntd)

- In MS-DOS, the interfaces and levels of functionality are not well separated. Application programs are able to access the basic I/O routines to write directly to the display and disk drives.
 - This causes entire system crashes when user programs fail.
 - Although MS-DOS was also limited by the hardware of its era which provided no dual mode and hardware protection, the designers of MS-DOS had no choice but to leave the base hardware accessible.
- 

Layered Approach

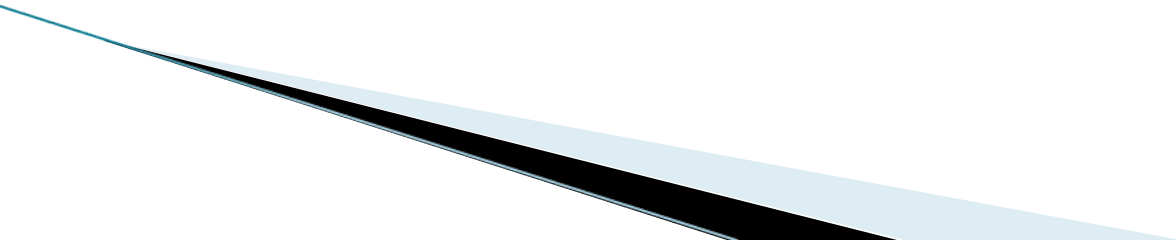
- With proper hardware support, operating systems can be broken into pieces that are smaller and more manageable. The operating system can then retain much greater control over the computer and over the applications that make use of that computer.
 - A system can be made modular in many ways. One method is the **layered approach**, in which the operating system is broken up into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.
- 

Layered Operating System



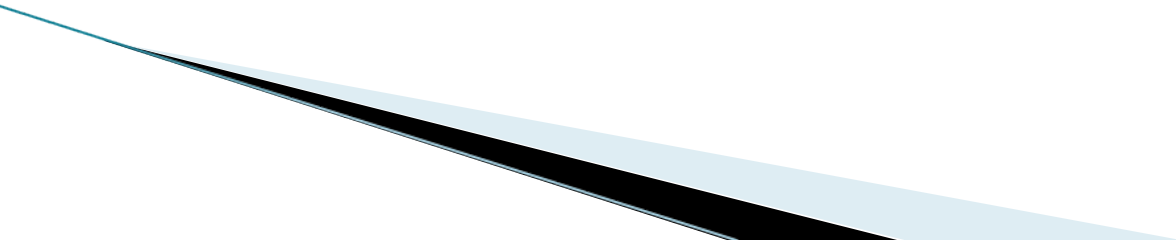
Layered Approach (cntd)

Advantages:

- The main advantage of the layered approach is simplicity of construction and **debugging**. The first layer can be debugged without any concern for the rest of the system, because, by definition, it uses only the basic hardware 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.
 - Each layer is implemented with only those operations provided by lower level layers. A layer does not need to know how these operations are implemented; it needs to know only what these operations do.
- 

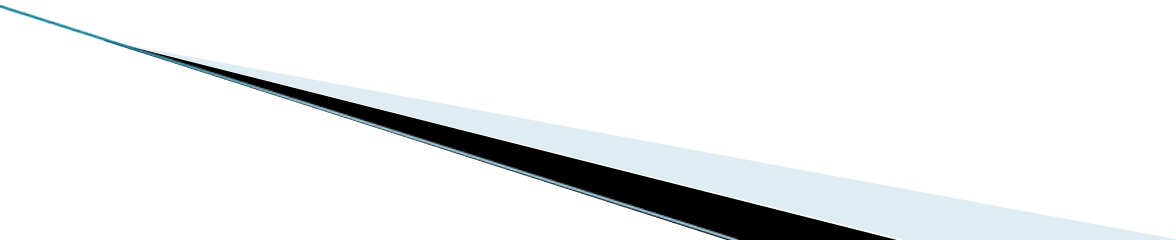
Layered Approach (cntd)

Disadvantages:

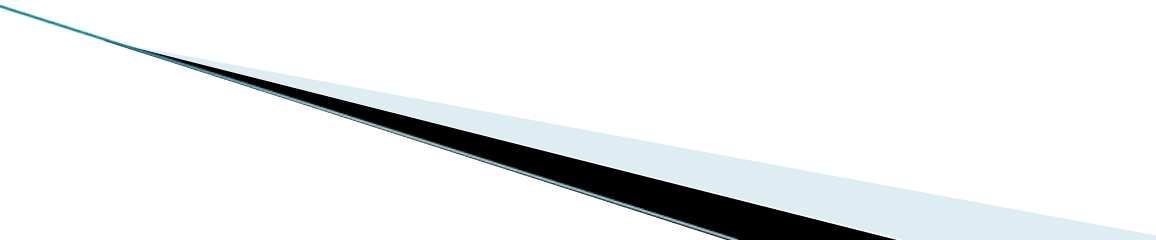
- The major difficulty with the layered approach involves appropriately defining the various layers due to **dependency** because a layer can use only lower-level layers, careful planning is necessary.
 - For example, the device driver for the harddisk(disk space used by virtual-memory algorithms) must be at a lower level than the memory-management routines, because memory management requires the ability to use the backing store.
 - A final problem with layered implementations is that they tend to be **less efficient** than other types.
- 

Layered Approach (cntd)

Disadvantages:

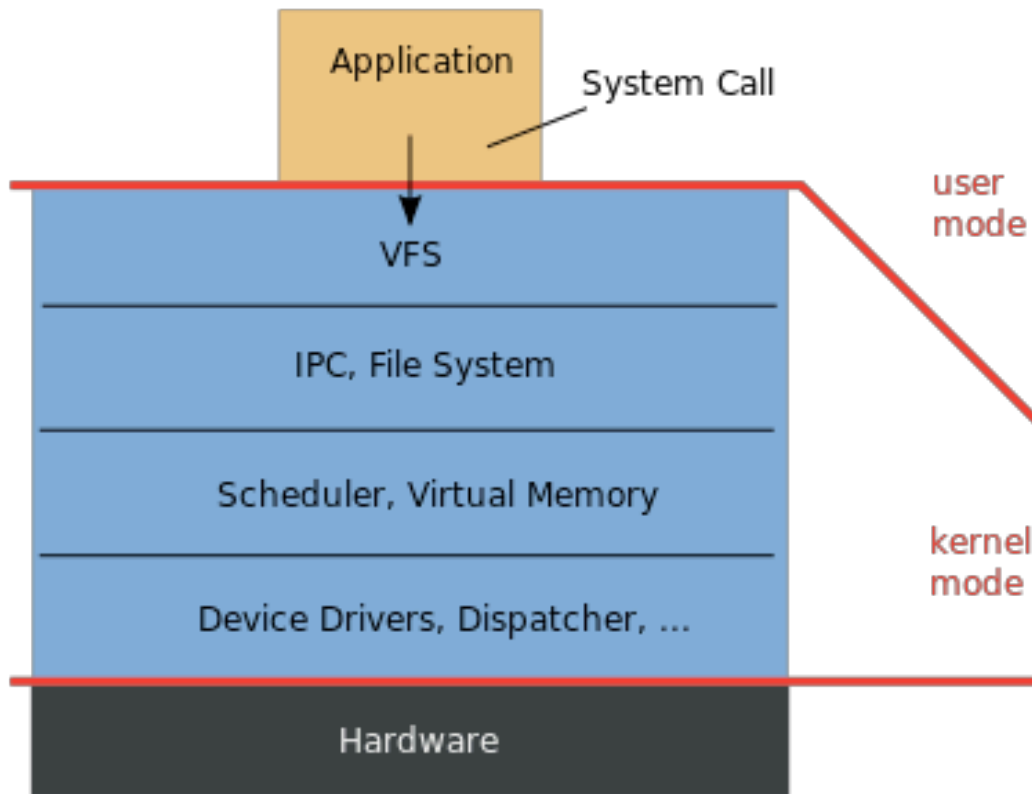
- For instance, when a user program executes an I/O operation, it executes a system call that is trapped to the I/O layer, which calls the memory-management layer, which in turn calls the CPU-scheduling layer, which is then passed to the hardware.
 - At each layer, the parameters may be modified, data may need to be passed, and so on. Each layer adds overhead to the system call; the net result is a system call that takes longer than does one on a non layered system.
- 

Microkernels

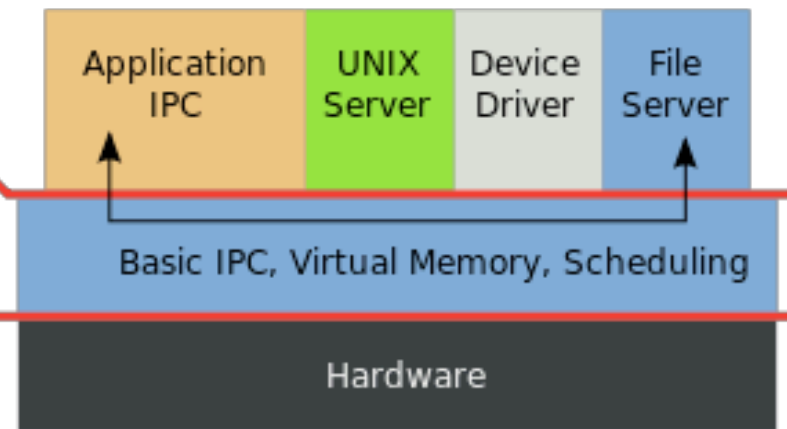
- Microkernel structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a **smaller kernel(OS)**.
 - There is no rule regarding which services should remain in the kernel and which should be implemented in user space. Typically, however microkernels provide minimal process and memory management, in addition to a communication facility.
 - Its main function is to provide a communication facility between the client program and the various services that are also running in user space. Communication is provided by message passing. For example, if the client program wishes to access a file, it must interact with the file server. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel.
- 

Microkernels

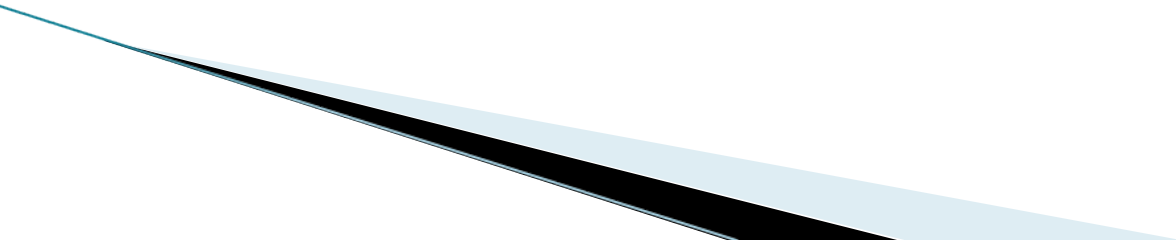
Monolithic Kernel based Operating System



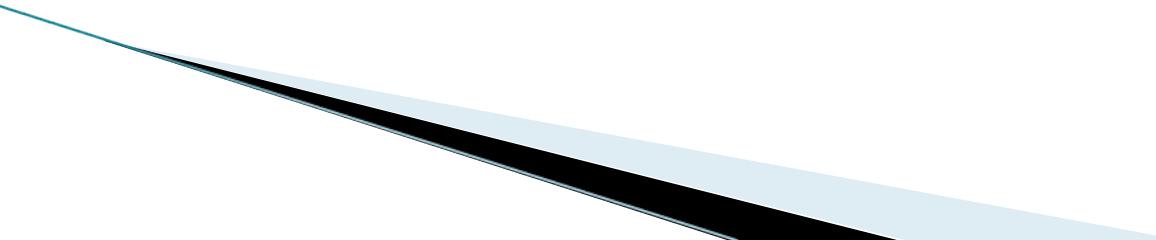
Microkernel based Operating System



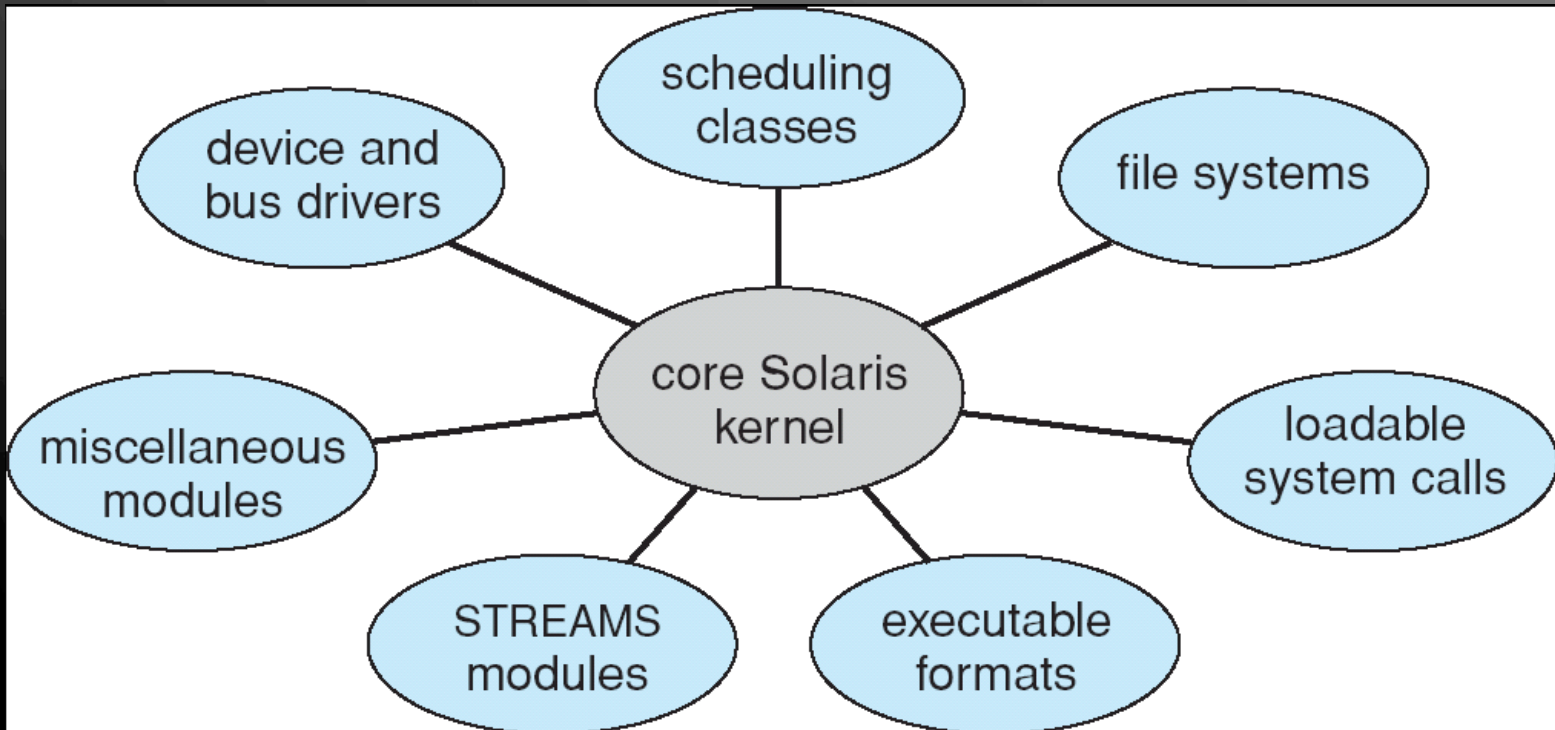
Microkernels (cntd)

- One benefit of the microkernel approach is **ease of extending the operating system**. All new services are added to user space and consequently do not require modification of the kernel.
 - The resulting operating system is easier to port from one hardware design to another.
 - The microkernel also provides more reliability, since most services are running as user—rather than kernel—processes. If a service fails, the rest of the operating system remains untouched.
- 

Modules

- ❑ Most modern operating systems implement kernel modules
 - Uses object-oriented approach
 - Each core component is separate
 - Each talks to the others over known interfaces
 - Each is loadable as needed within the kernel
 - ❑ Overall, similar to layers but with more flexible. For example, the Solaris operating system structure is organized around a core kernel with seven types of loadable kernel modules:
 - ❑ 1. Scheduling classes
 - ❑ 2. File systems
 - ❑ 3. Loadable system calls
 - ❑ 4. Executable formats
 - ❑ 5. STREAMS modules
 - ❑ 6. Miscellaneous
 - ❑ 7. Device and bus drivers
- 

Solaris Modular Approach



Modules (cntd)

- Such a design allows the kernel to provide core services yet also allows certain features to be implemented dynamically. For example, device drivers for specific hardware can be added to the kernel **as loadable modules**.
 - The overall result resembles a layered system in that each kernel section has defined, protected interfaces; but it is more **flexible** than a layered system in that any module can call any other module.
 - Also the approach is like the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules; but it is more **efficient**, because modules do not need to invoke message passing in order to communicate.
- 