L6: Operating system structure: simple structure, layered , micro kernel, modular approach

Dr. Rajashree Dash

Associate Professor,
Department of CSE,
ITER, Siksha O Anusandhan Deemed to be University

Outline

Operating system structure

- General-purpose OS is very large and complex program.
- It should be structured carefully to function properly and be modified easily.
- One common approach is to partition the task into small components rather having one monolithic system. Each of the modules should be a well defined portion of the system, with carefully defined inputs, outputs and functions. Various ways to structure OS are:
  - Simple monolithic structure
  - Layered
  - Microkernel
  - Modular

Simple structure: MS-DOS

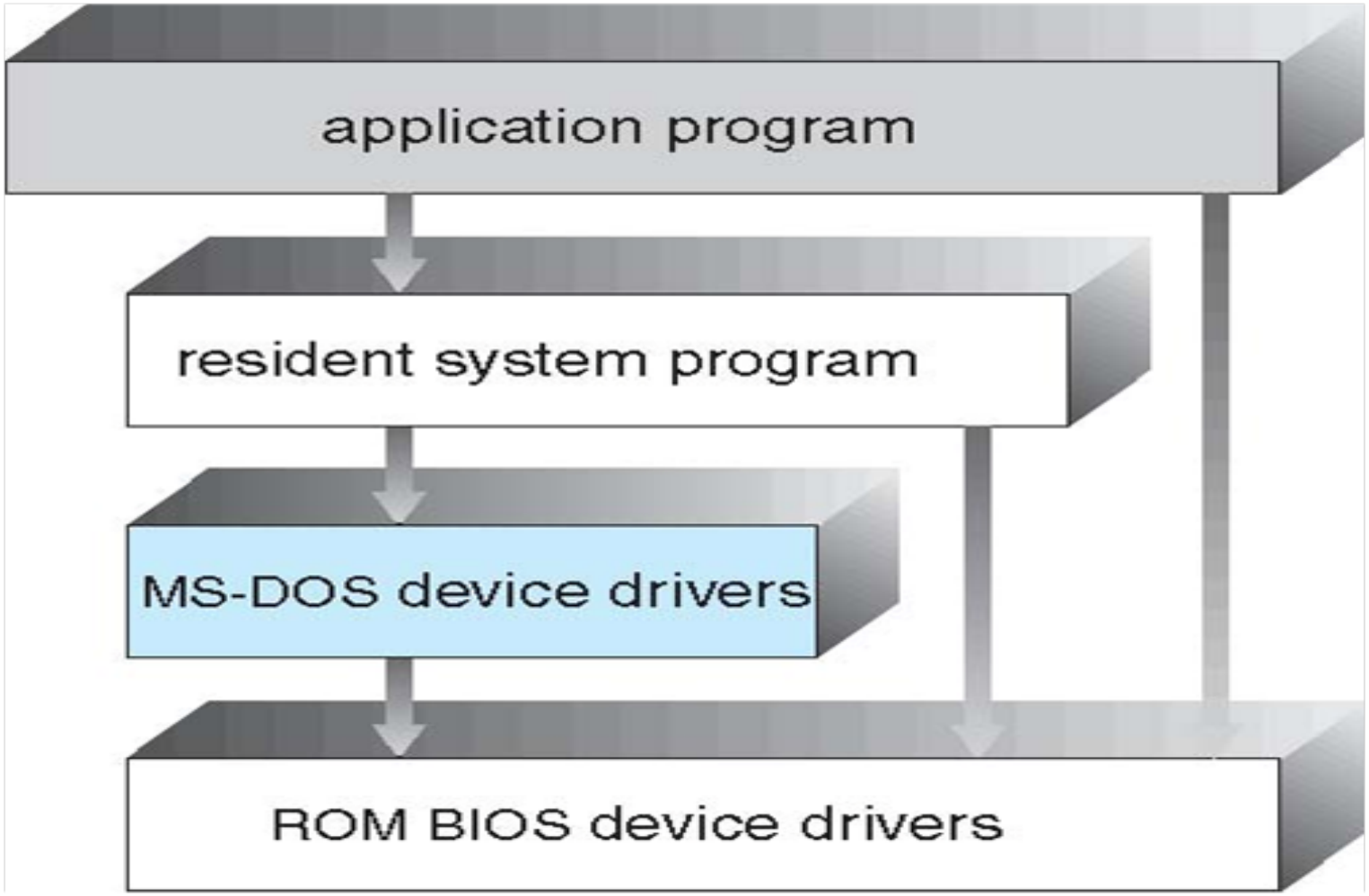

Figure: Simple Structure - MS-DOS

Simple structure: MS-DOS

- MS-DOS – written to provide the most functionality in the least space.
- Not divided into modules.
- Here 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. Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system crashes, when user programs fail.
- No dual mode and no hardware protection.

Simple structure: UNIX

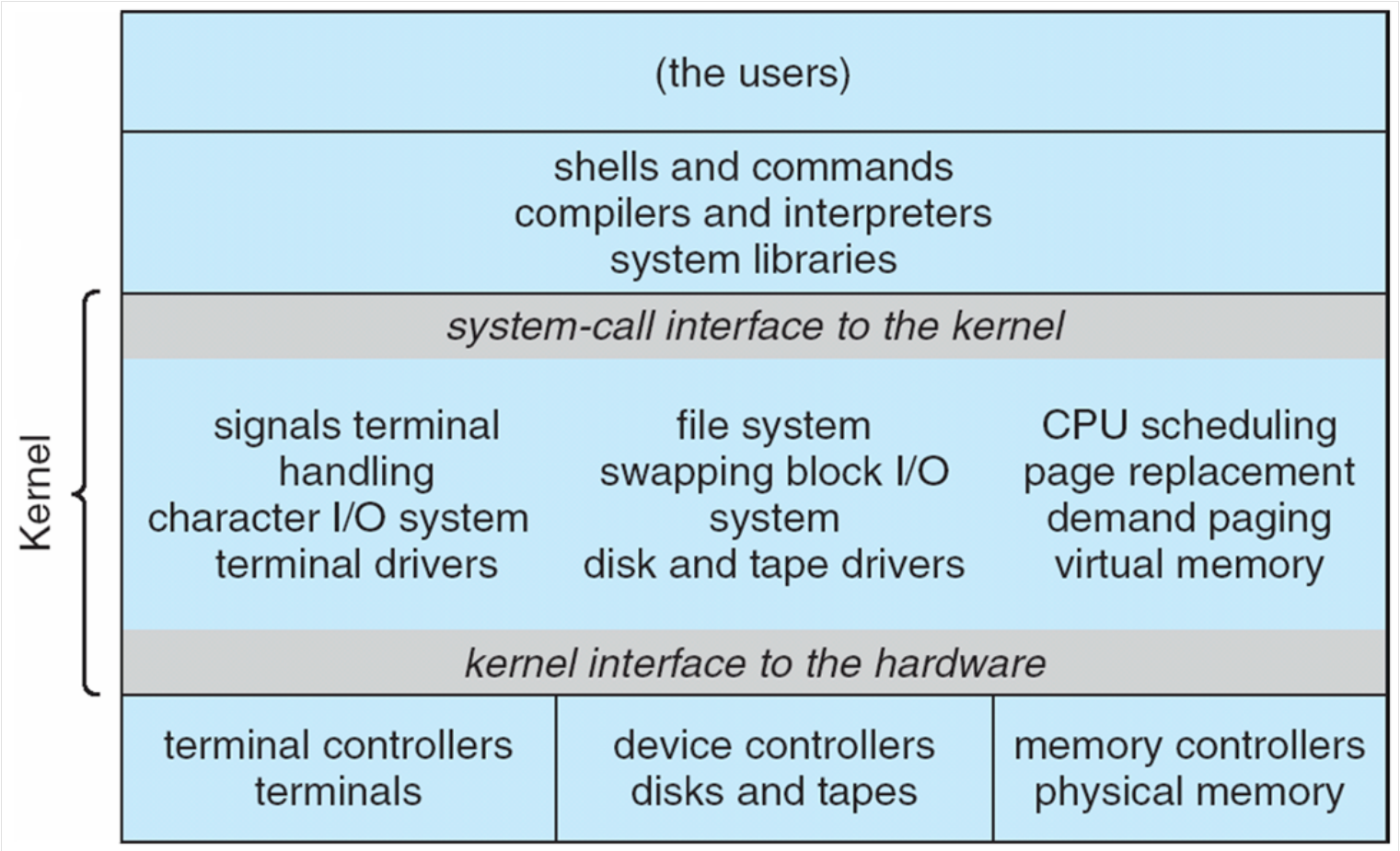

Figure: Simple Structure - UNIX

Simple structure: UNIX

- UNIX – initially was limited by hardware functionality, the original UNIX operating system had limited structuring.
- The UNIX OS consists of two separable parts:
  - Systems programs
  - The kernel

    • Consists of everything below the system-call interface and above the physical hardware.

    • Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

Layered Structure

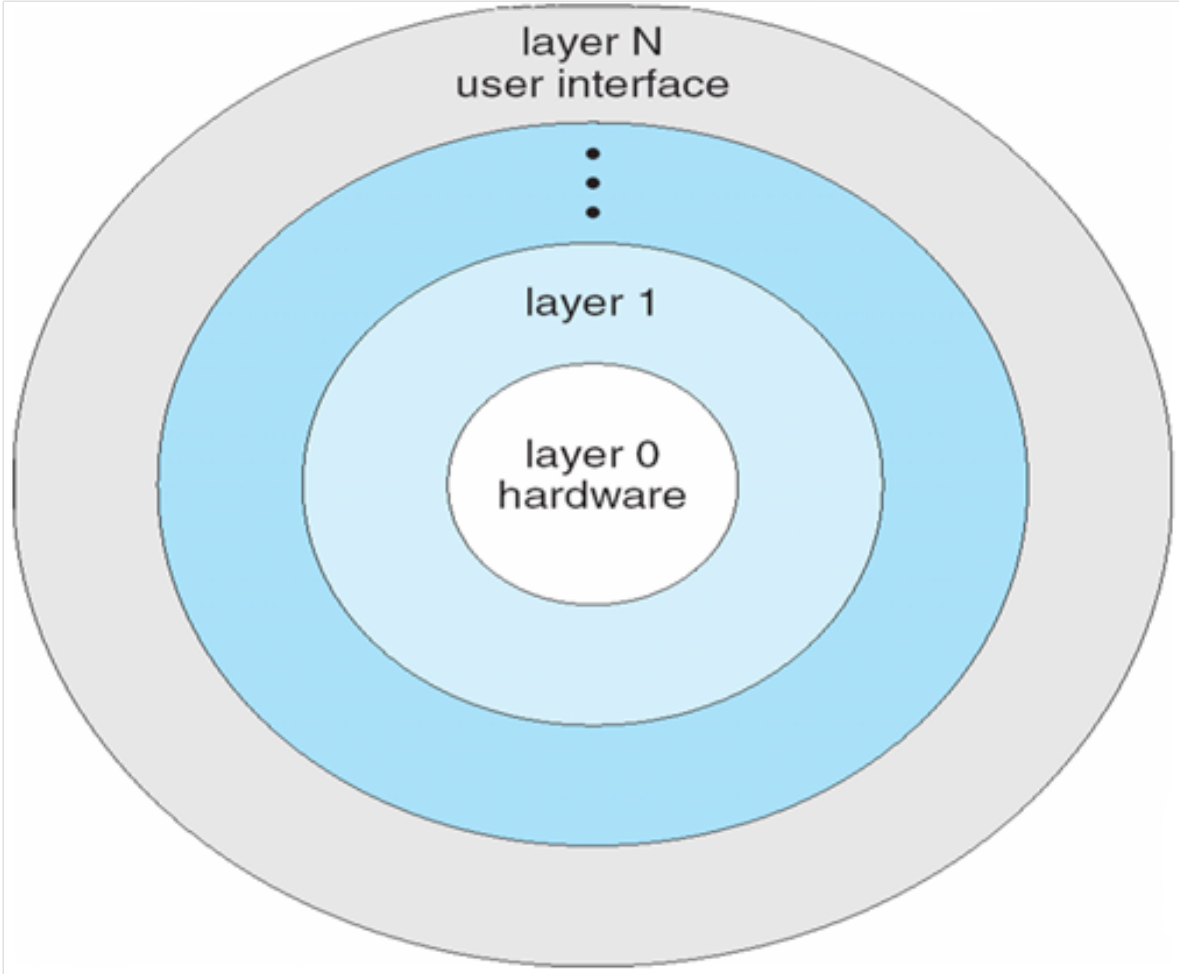

Figure: Layered Structure

Layered Structure

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- Operating systems can be broken into pieces that are smaller and more appropriate.
- The operating system can then retain much greater control over the computer and over the applications that make use of that computer. Implementers have more freedom in changing the inner workings of the system and in creating modular operating systems.

Advantages of Layered Structure

- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers. This approach simplifies debugging and system verification.
- Each layer is implemented only with 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. Hence, each layer hides the existence of certain data structures, operations, and hardware from higher-level layer.

Disadvantages of Layered Structure

- The major difficulty with the layered approach involves appropriately defining the various layers.
- Each layer adds overhead to the system call. The net result is a system call that takes longer than does one on a nonlayered system. (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.)

Microkernel Structure

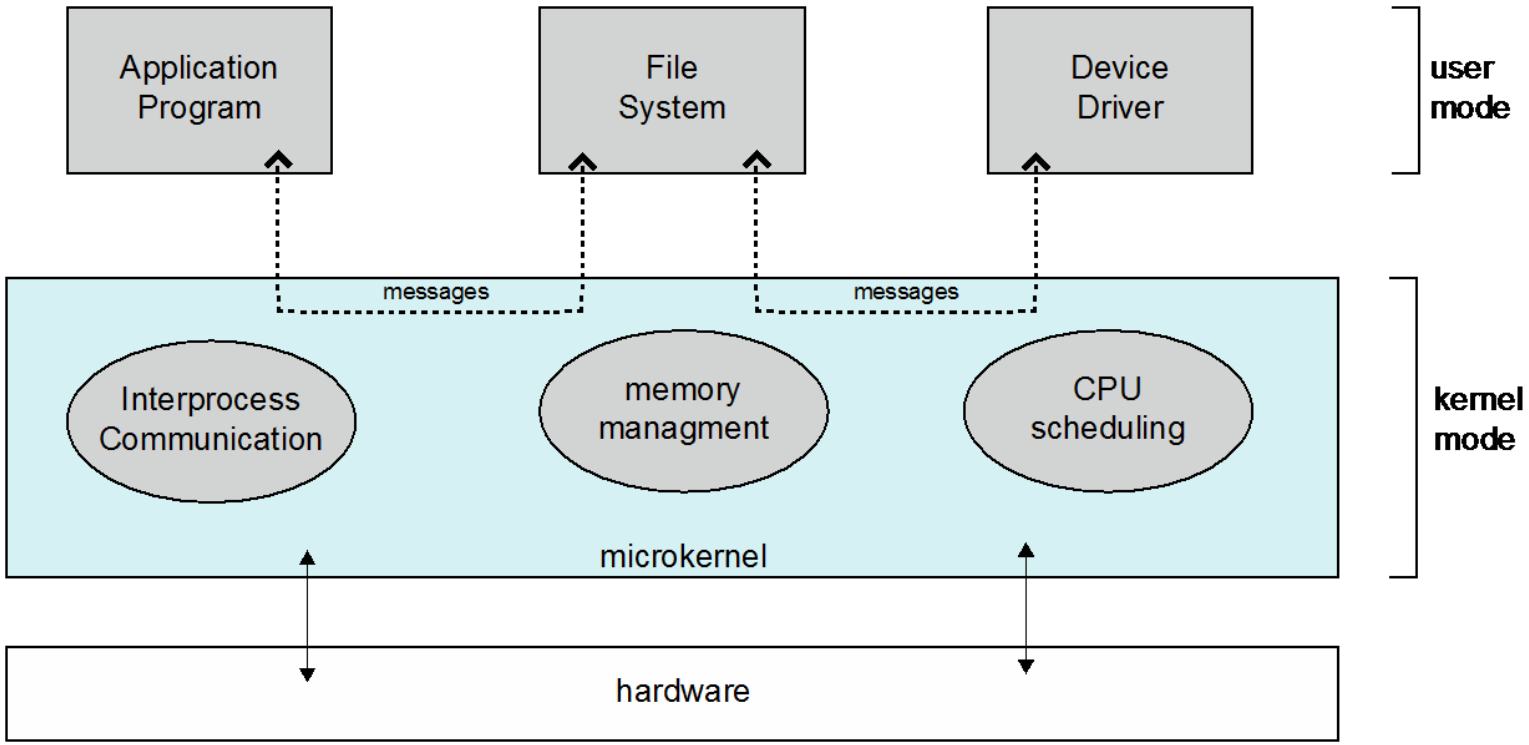

Figure: Microkernel Structure

Microkernel Structure

- Mach that modularized the kernel using the microkernel approach.
- It 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.
- Microkernel provides minimal process and memory management, in addition to a communication facility.
- Communication takes place between user modules using message passing.

Microkernel Structure

- Benefits:
  - Easier to extend a microkernel. New services are added to user space and consequently do not require modification of the kernel.
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Disadvantage:

  – Performance overhead of user space to kernel space communication.

Modular Structure

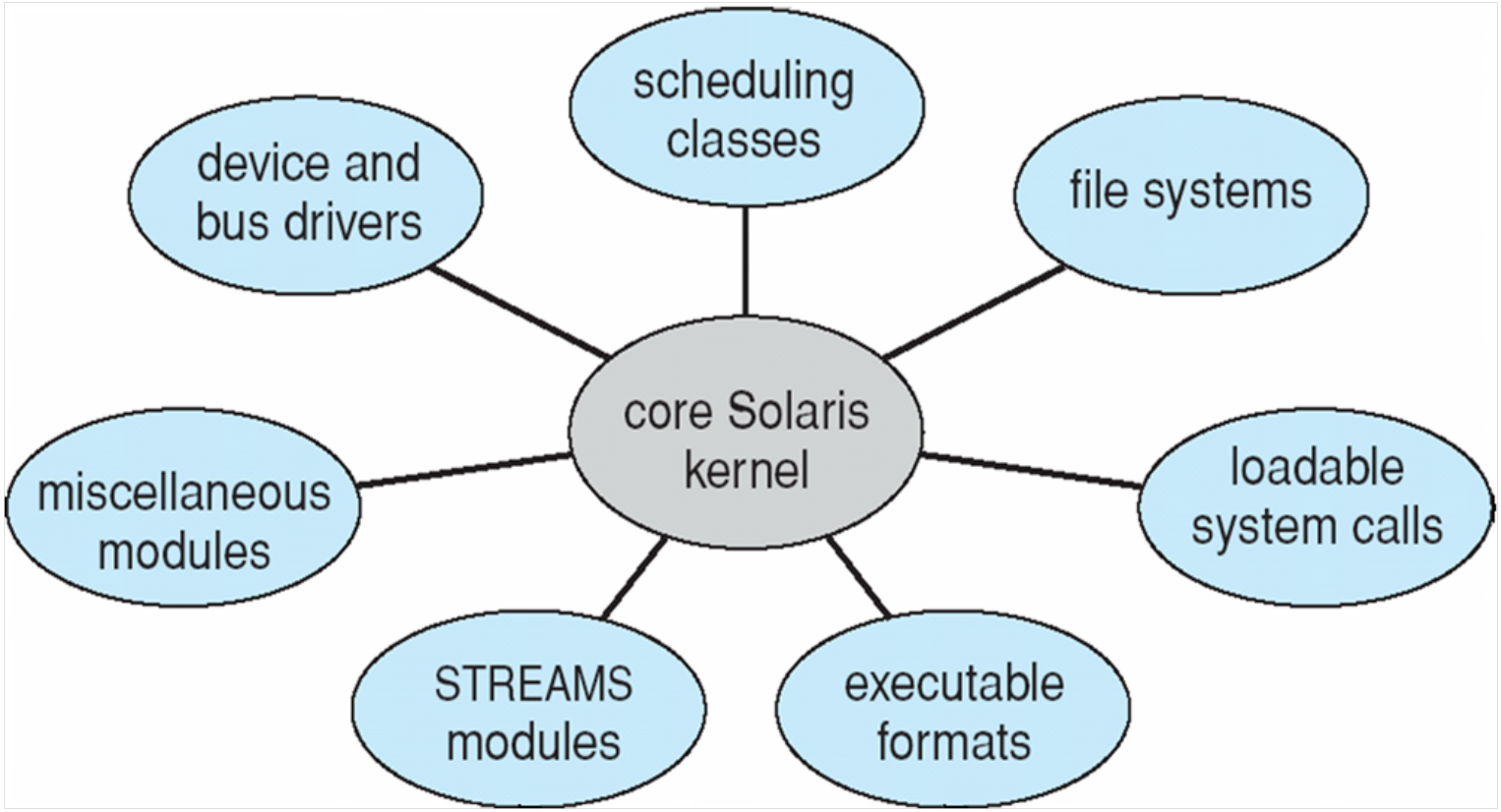

Figure: Modular Structure

Modular Structure

- Many modern operating systems implement loadable kernel modules. The kernel has a set of core components and links in additional services via modules, either at boot time or during run time. This type of design is common in modern implementations of UNIX , such as Solaris, Linux, and Mac OS X , as well as Windows.
- Core services are provided for the kernel and other services are implemented dynamically, as the kernel is running.
- Linking services dynamically is preferable to add new features directly to the kernel, which would require recompiling the kernel every time a change was made.

Modular Structure

- Like a layered system kernel section has defined, protected interfaces; but it is more flexible than a layered system, because any module can call any other module.
- Like microkernel approach, 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.
- Benefits:
  - Each core component is separate.
  - Each talks to the others over known interfaces.
  - Each is loadable as needed within the kernel.

Hybrid approach

- Most modern operating systems combine different structures, resulting in hybrid systems that address performance, security, and usability issues.
- Both Linux and Solaris 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 is largely monolithic as well (again primarily for performance reasons), but it retains some behaviour typical of microkernel systems, including providing support for separate subsystems (known as operating-system personalities) that run as user-mode processes. Windows systems also provide support for dynamically loadable kernel modules.