Architecture of Operating system

1

1

Operating System Architecture

Simple

Layered

Microkernels

Modules

Hybrid

2

2

Monolithic Architecture

OSs were developed to meet various requirements, such as CPU busyness, multiple jobs in batch systems, and user friendliness.

However, the initial architecture was not efficient and consisted of few modules due to limited functionality. The kernel only added all functionalities, allowing efficient intercommunication between modules.

This development process was not planned and led to unplanned OS development.

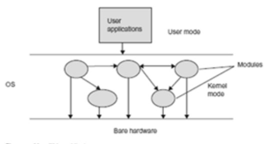


Fig. Monolithic architecture

3

3

Monolithic Architecture

Multi-programming expanded the size of the operating system (OS), leading to a complex structure where every module directly accesses hardware.

Therefore, there is a large gap in understanding the operations at OS level and machine level, as the OS consists of algorithms for process allocation and scheduling, while hardware operations are performed at machine instructions.

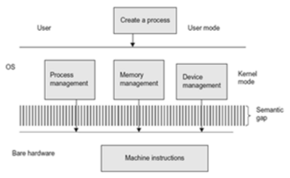


Fig. Semantic gap between the OS and hardware

4

4

Monolithic Architecture

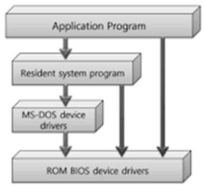
MS-DOS

It was original designed and implemented by a few people.

It was written to provide the most functionality in the least space.

In MS-DOS, the interfaces and levels of functionality are not well separated.

Vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail.



5

5

Monolithic Architecture

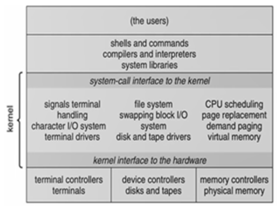
UNIX

Like MS-DOS, UNIX initially was limited by hardware functionality. It consists of two separable parts: the kernel and the system programs.

The kernel is further separated into a series of interface and device drivers.

The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.

This monolithic structure was difficult to implement and maintain.



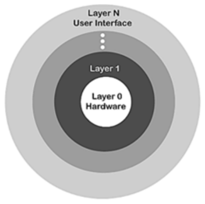
6

6

Layer Architecture

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.

With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



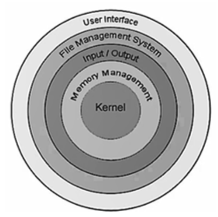
7

Layer Architecture

Less efficient.

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 net result is a system call that takes longer than does one on a non-layerd system.



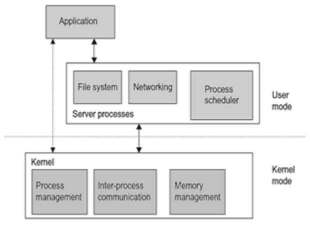
8

Microkernel Architecture

Removes all nonessential components from the kernel and implements them as system and user-level programs. The result is a small kernel.

Microkernel provide minimal process and memory management.

The main function of the microkernel is to provide communication between the client program and the services running in user space.



9

Microkernel Architecture

Easier to port from one hardware design to another.

The microkernel also provides more security and reliability.

If a service fails, the rest of the OS remains untouched

10

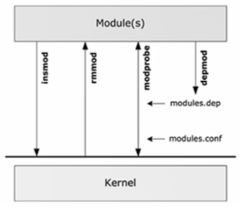
Modules Architecture

The best current methodology for operating system design involves using loadable kernel modules.

The kernel har a set of core components and links in additional services via modules, either at boot time or during run time.

The lernel to provide core services while other services are implemented dynamically, as the kernel is running.

Linking services dynamically is preferable to adding new features directly to the kernel, which would require recompiling the kernel every time a change was made.

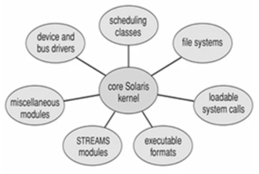


11

Modules Architecture

More flexible than a layered system, because any module can call any other module.

More efficient than a microkernel, because modules do not need to invoke message passing in order to communication.



12

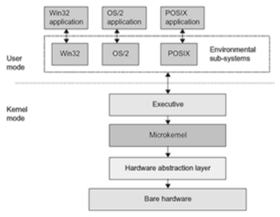
Hybrid kernel Architecture

In practice, very few operating systems adopt a single, strictly defined structure.

They combine different structures, resulting in hybrid systems that address performance, security, and usability issues.

Three hybrid systems:

- Apple Mac OS X
- iOS
- Android



13

13