MONOLITHIC KERNEL

Kernel

Monolithic Kernel

Monolithic Kernels vs. Microkernels

Kernel

kernel is the core piece of most operating systems. It is a piece of software responsible for the communication between hardware and software components. As a basic component of an operating system, a kernel provides abstraction layers for hardware, especially for memory, processors and communication between hardware and software. It also provides software facilities to userland applications such as process abstractions, interprocess communication and system calls.

Tasks of Kernel

The definition of a kernel is basically a house-keeping program that runs at the highest privilege level, manages the computer's resources and allows other programs to run. This involves tasks such as:

- Memory management
- Process management
- I/O management

Furthermore, a kernel must provide userland programs a way to access these services through some kind of system calls.

Types of Kernel

There are four broad categories of kernels:

- *Monolithic* kernels provide rich and powerful abstractions of the underlying hardware.
- *Microkernels* provide a small set of simple hardware abstractions and use applications called servers to provide more functionality.

Types of Kernel

- *Hybrid kernels* (modified microkernels) are much like pure microkernels, except that they include some additional code in kernelspace in order to increase performance.

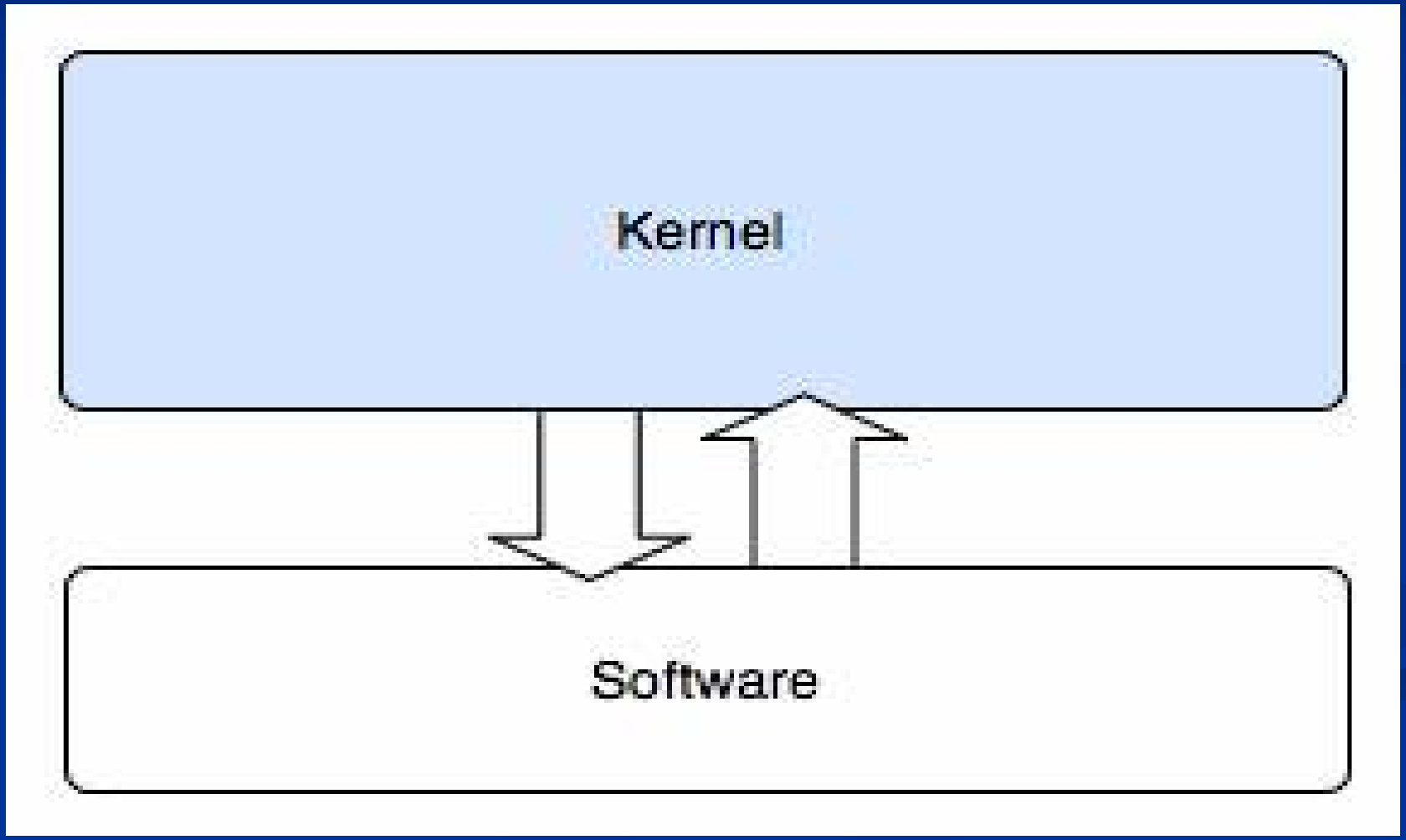
- *Exokernels* provide minimal abstractions, allowing low-level hardware access. In exokernel systems, library operating systems provide the abstractions typically present in monolithic kernels.

Monolithic Kernel

A monolithic kernel defines a high-level virtual interface over the hardware, with a set of *primitives* or *system calls* to implement operating system services such as process management, *concurrency*, and *memory management* in several *modules* that run in *supervisor mode*.

In a monolithic kernel, all OS services run along with the main kernel thread, thus also residing in the same memory area. This approach provides rich and powerful hardware access.

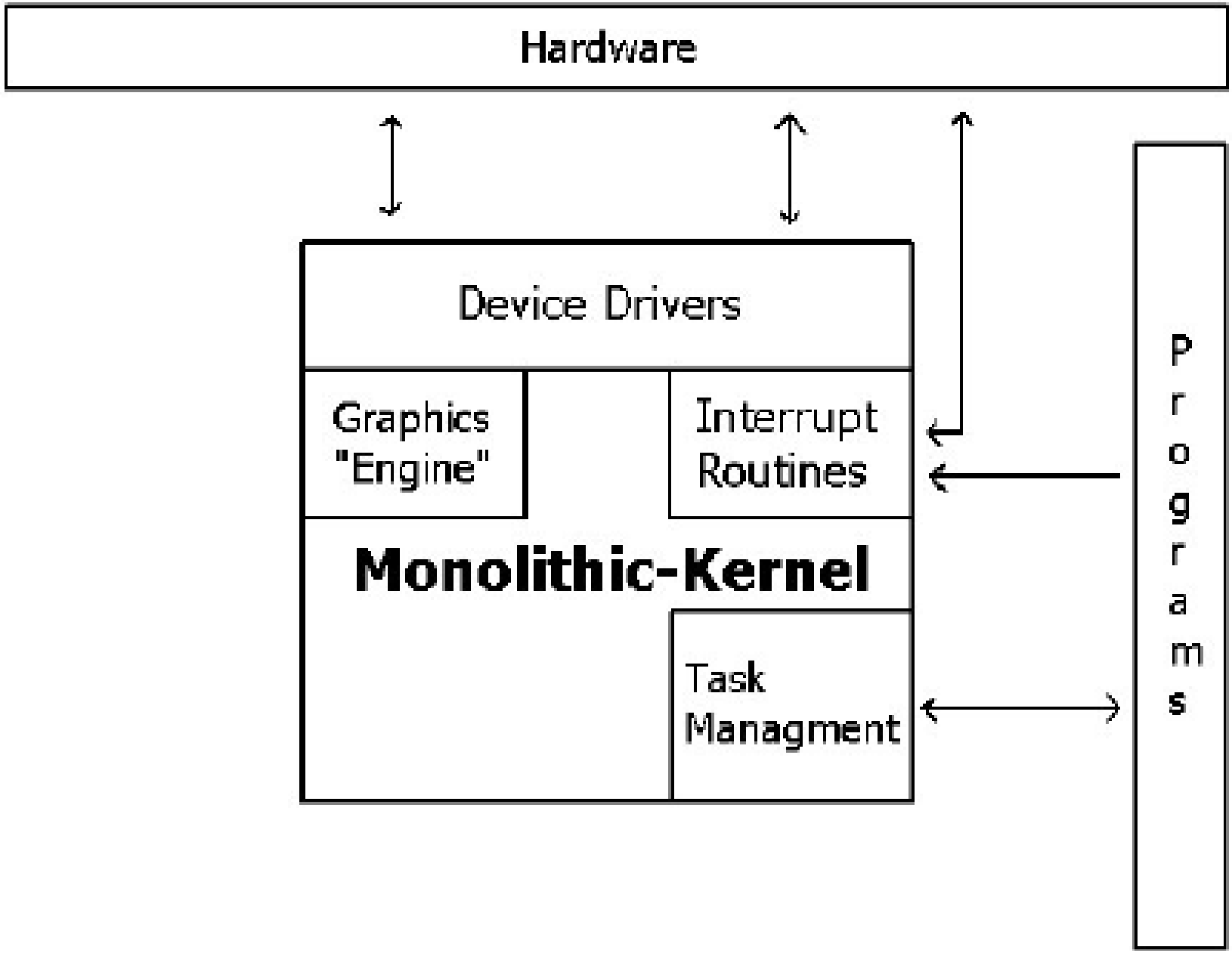
Monolithic Kernel

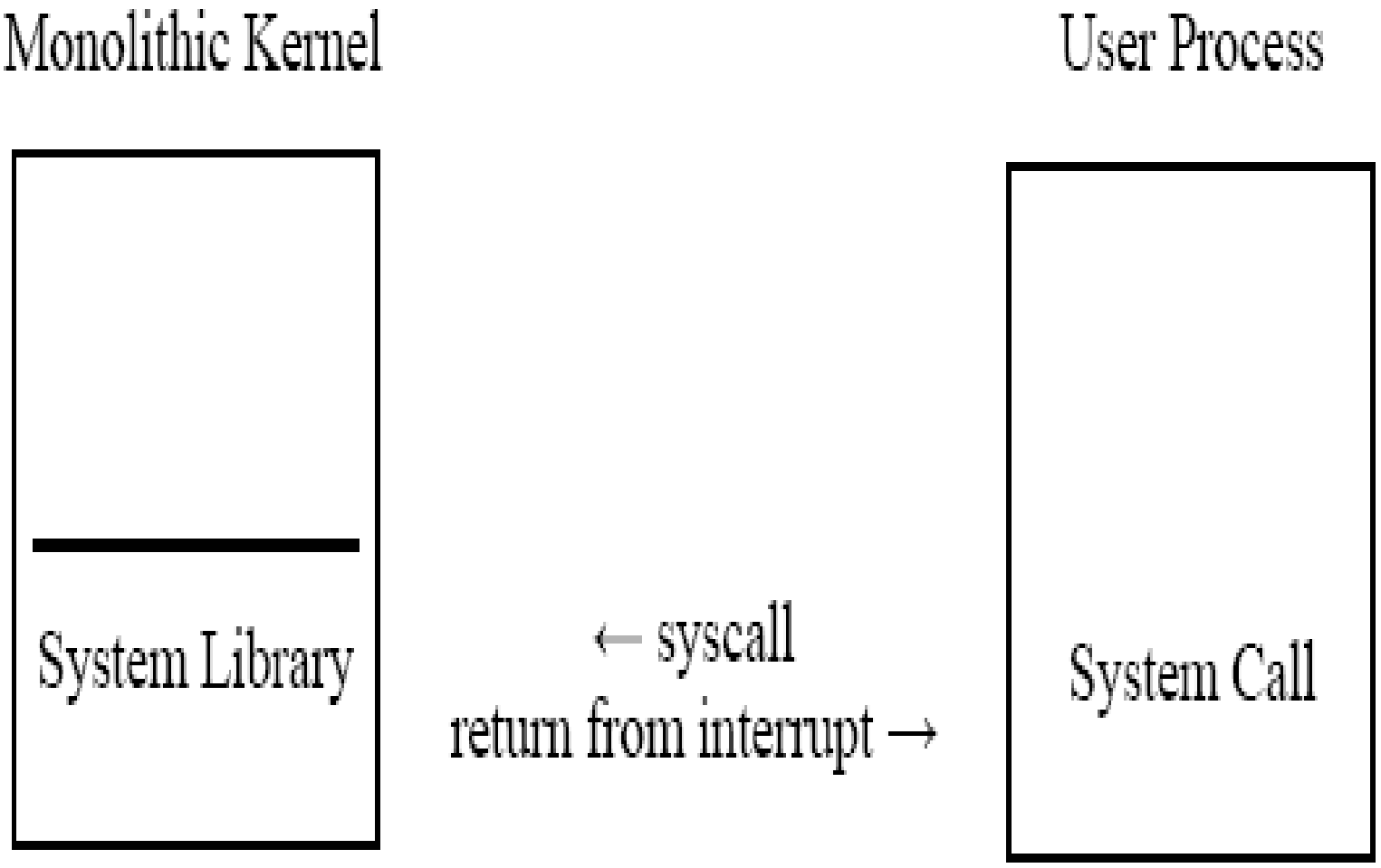


Monolithic Kernel

Even if every module servicing these operations is separate from the whole, the code integration is very tight and difficult to do correctly, and, since all the modules run in the same address space, a bug in one module can bring down the whole system. However, when the implementation is complete and trustworthy, the tight internal integration of components allows the low-level features of the underlying system to be effectively utilized, making a good monolithic kernel highly efficient. In a monolithic kernel, all the processes such as the filesystem management run in an area called the kernel mode.

A Monolithic-Kernel



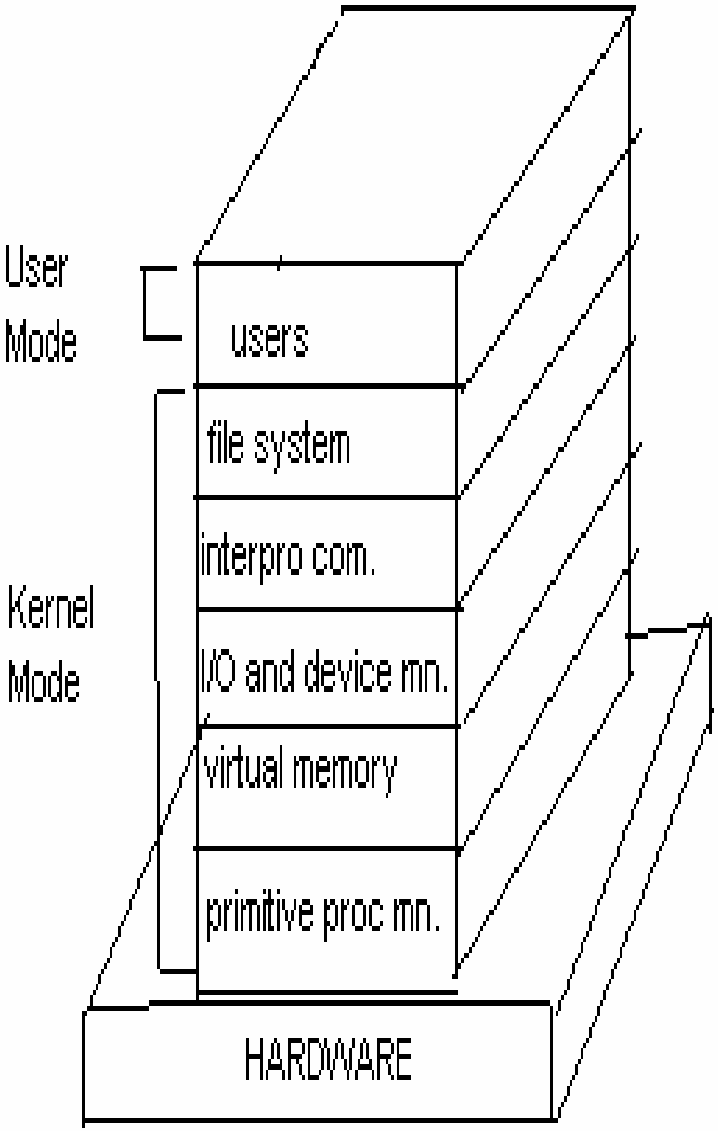


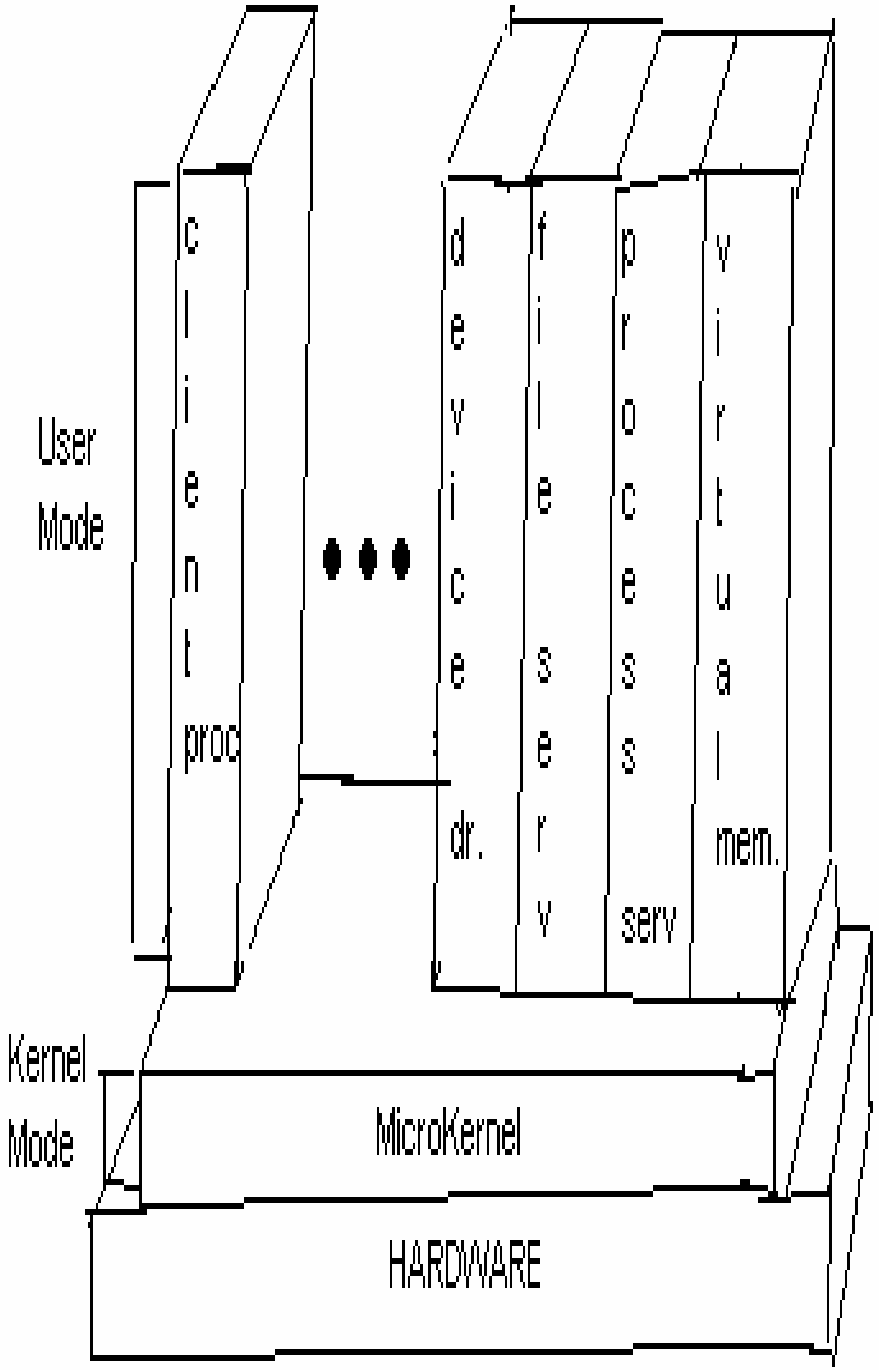
Problem: all system calls execute in kernel mode, syscall is relatively expensive.

Monolithic Kernel vs. Microkernel

Monolithic kernels tend to be easier to design correctly, and therefore may grow more quickly than a microkernel-based system. However, a bug in a monolithic system usually crashes the entire system; this would not happen in a microkernel with servers running apart from the main thread. Monolithic kernel proponents reason that incorrect code doesn't belong in a kernel, and microkernels offer little advantage for correct code. There are success stories in both camps.

Microkernels are often used in embedded robotic or medical computers because most of the OS components reside in their own private, protected memory space. This is impossible with monolithic kernels, even with modern module-loading ones. However, the monolithic model tends to be more efficient through the use of shared kernel memory, rather than the slower Inter-process communication characteristic of microkernel designs.


User
Mode
users
file system
interpro com.
Kernel
Mode
I/O and device mn.
virtual memory
primitive proc mn.
HARDWARE


User
Mode
client proc
device dr.
file serv
process serv
virtual mem.
Kernel
Mode
MicroKernel
HARDWARE

Monolithic Kernel Examples

- Traditional *Unix* kernels, such as the kernels of the *BSDs* and *Solaris*
- *Linux kernel*
- Some educational kernels, such as *Agnix*
- *MS-DOS, Microsoft Windows 9x series (Windows 95, Windows 98 and Windows 98SE) and Windows Me.*
- *Mac OS* kernel, up until *Mac OS 8#Mac OS 8.6*
- *OpenVMS*

Summary

Monolithic systems are easier to design and implement than other solutions, also being very efficient if well-written. The main disadvantage of monolithic kernels is the dependency between system components - a bug might crash the entire system - and the fact that large kernels also become difficult to maintain.

References

- http://www.cse.psu.edu
- http://wikipedia.org
- Stallings, William (1998). Operating Systems: Internals & Design Principles. New Jersey, Printice Hall.