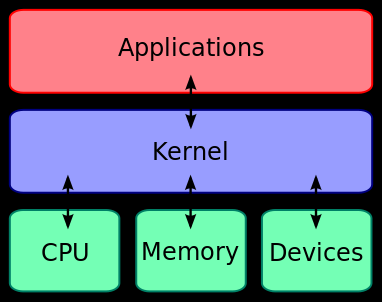
**Practical - 6**

**Aim: Case Study to understand the basic Kernels (monolithic and micro) and also the two basic modes of execution***.*

# INTRODUCTION

In [computing](http://en.wikipedia.org/wiki/Computing), the **kernel** is a [computer program](http://en.wikipedia.org/wiki/Computer_program) that manages [input/output](http://en.wikipedia.org/wiki/Input/output) requests from [software](http://en.wikipedia.org/wiki/Software) and translates them into [data processing](http://en.wikipedia.org/wiki/Data_processing) instructions for the [central processing unit](http://en.wikipedia.org/wiki/Central_processing_unit) and other [electronic components](http://en.wikipedia.org/wiki/Electronic_component) of a [computer](http://en.wikipedia.org/wiki/Computer). The kernel is a fundamental part of a modern computer's [operating system](http://en.wikipedia.org/wiki/Operating_system).

When a computer program (in this case called a [*process*](http://en.wikipedia.org/wiki/Process_%28computing%29)) makes requests of the kernel, the request is called a [system call](http://en.wikipedia.org/wiki/System_call). Various kernel designs differ in how they manage system calls ([time-sharing](http://en.wikipedia.org/wiki/Time-sharing)) and [resources](http://en.wikipedia.org/wiki/Resource_%28computer_science%29). For example, a [monolithic kernel](http://en.wikipedia.org/wiki/Monolithic_kernel) executes all the operating system [instructions](http://en.wikipedia.org/wiki/Instruction_set) in the same [address space](http://en.wikipedia.org/wiki/Address_space) to improve the performance of the system. A [microkernel](http://en.wikipedia.org/wiki/Microkernel) runs most of the operating system's [background process](http://en.wikipedia.org/wiki/Background_process) in [user space](http://en.wikipedia.org/wiki/User_space), to make the operating system more [modular](http://en.wikipedia.org/wiki/Modular_programming) and, therefore, easier to maintain.



*Fig. 6.1 Kernel Architected*

# TYPES OF BASIC KERNALS

**1. Monolithic Kernel**

**2. Micro Kernel**

# MONOLITHIC KERNAL

A **monolithic kernel** is an operating system architecture where the entire operating system is working in [kernel space](http://en.wikipedia.org/wiki/Kernel_space) and is alone in [supervisor mode](http://en.wikipedia.org/wiki/Supervisor_mode). The monolithic model differs from other operating system architectures (such as the [microkernel](http://en.wikipedia.org/wiki/Microkernel) architecture).in that it alone defines a high-level virtual interface over computer hardware. A set of primitives or [system calls](http://en.wikipedia.org/wiki/System_call) implement all operating system services such as [process](http://en.wikipedia.org/wiki/Process_%28computing%29) management, [concurrency](http://en.wikipedia.org/wiki/Concurrency_%28computer_science%29), and [memory management](http://en.wikipedia.org/wiki/Memory_management). Device drivers can be added to the kernel as [modules](http://en.wikipedia.org/wiki/Module_%28programming%29).

# Microkernel designs

Examples

* AIX
* AmigaOS
* Amoeba
* Chorus
* Mach
* Minix (remember Tannenbaum vs. Torvalds?)
* QNX
* Symbian OS

# Microkernel designs

* Design goals
* use a very simple abstraction over hardwarethread manage address spaces
* interprocess communication
* reduce the functionality in supervisor mode
* move specific drivers to user-space
* isolate individual network services in a separate context

# Benefit of microkernels

* Essential to trusted computing concepts
* Provides a small, controlled operating system core
* Controls the hardware and resources
* Holds initial ownership of all resources
* Delegates individual resources to trusted processes
* Isolates privileges and resources
* Can provide service level guarantees

# MICRO KERNEL

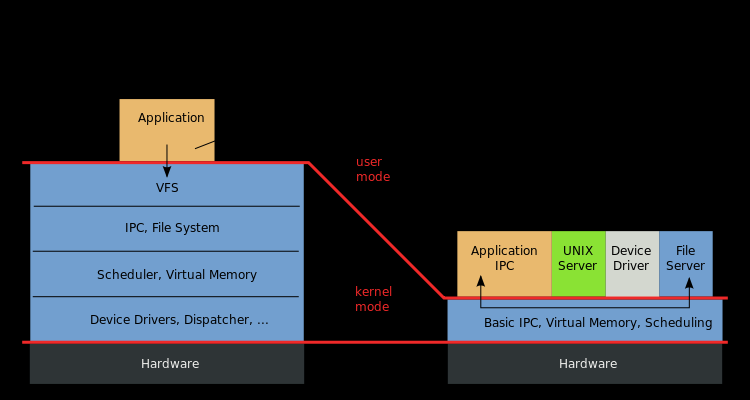
In [computer science](http://en.wikipedia.org/wiki/Computer_science), a **microkernel** (also known as μ-kernel or Samuel kernel) is the near-minimum amount of software that can provide the mechanisms needed to implement an [operating system](http://en.wikipedia.org/wiki/Operating_system) (OS). These mechanisms include low-level [address space](http://en.wikipedia.org/wiki/Address_space) management, [thread](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) management, and [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication) (IPC). If the hardware provides multiple [rings](http://en.wikipedia.org/wiki/Hierarchical_protection_domains) or [CPU modes](http://en.wikipedia.org/wiki/CPU_modes), the microkernel is the only software executing at the most privileged level (generally referred to as [supervisor or kernel mode](http://en.wikipedia.org/wiki/Kernel_mode)). Traditional operating system functions, such as [device drivers](http://en.wikipedia.org/wiki/Device_driver), [protocol stacks](http://en.wikipedia.org/wiki/Protocol_stack) and [file systems](http://en.wikipedia.org/wiki/File_system), are removed from the microkernel to run in [user space](http://en.wikipedia.org/wiki/User_space). In source code size, microkernels tend to be under 10,000 lines of code, as a general rule. [MINIX](http://en.wikipedia.org/wiki/MINIX)'s kernel, for example has fewer than 6,000 lines of code.

Microkernels were developed in the 1980s as a response to changes in the computer world, and to several challenges adapting existing "mono-kernels" to these new systems. New device drivers, protocol stacks, file systems and other low-level systems were being developed all the time. This code was normally located in the monolithic kernel, and thus required considerable work and careful code management to work on. Microkernels were developed with the idea that all of these services would be implemented as user-space programs, like any other, allowing them to be worked on monolithically and started and stopped like any other program. This would not only allow these services to be more easily worked on, but also separated the kernel code to allow it to be finely tuned without worrying about unintended side effects. Moreover, it would allow entirely new operating systems to be "built up" on a common core, aiding OS research.

Microkernels were a very hot topic in the 1980s when the first usable [local area networks](http://en.wikipedia.org/wiki/Local_area_network) were being introduced. The same mechanisms that allowed the kernel to be distributed into user space also allowed the system to be distributed across network links. The first microkernels, notably [Mach](http://en.wikipedia.org/wiki/Mach_%28kernel%29), proved to have disappointing performance, but the inherent advantages appeared so great that it was a major line of research into the late 1990s. However, during this time the speed of computers grew greatly in relation to networking systems, and the disadvantages in performance came to overwhelm the advantages in development terms. Many attempts were made to adapt the existing systems to have better performance, but the overhead was always considerable and most of these efforts required the user-space programs to be moved back into the kernel. By 2000, most large-scale (Mach-like) efforts had ended, although [Open Step](http://en.wikipedia.org/wiki/OpenStep) used an adapted Mach kernel called [XNU](http://en.wikipedia.org/wiki/XNU), which is now used in the OS known as [Darwin](http://en.wikipedia.org/wiki/Darwin_%28operating_system%29), which is the open source part of [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X). As of 2012, the Mach-based [GNU Hurd](http://en.wikipedia.org/wiki/GNU_Hurd) is also functional and its inclusion in testing versions of [Arch Linux](http://en.wikipedia.org/wiki/Arch_Linux) and [Debi an](http://en.wikipedia.org/wiki/Debian) is in progress.

Although major work on microkernels had largely ended, experimenters continued development. It has since been shown that many of the performance problems of earlier designs were not a fundamental requirement of the concept, but instead due to the designer's desire to use single-purpose systems to implement as many of these services as possible. Using a more pragmatic approach to the problem, including [assembly code](http://en.wikipedia.org/wiki/Assembly_code) and relying on the processor to enforce concepts normally supported in software led to a new series of microkernels with dramatically improved performance.

Microkernels are closely related to [exokernels](http://en.wikipedia.org/wiki/Exokernel). They also have much in common with [hypervisors](http://en.wikipedia.org/wiki/Hypervisor), but the latter make no claim to minimality and are specialized to supporting [virtual machines](http://en.wikipedia.org/wiki/Virtual_machine); indeed, the [L4 microkernel](http://en.wikipedia.org/wiki/L4_microkernel) frequently finds use in a hypervisor capacity.



*Fig. Micro Kernel*