TYPES OF OPERATING SYSTEM KERNELS

Cristian ULMANU

Universitatea Tehnică a Moldovei

Abstract: *A kernel is a central component of an operating system. It acts as an interface between the user applications (software) and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware components (CPU, disk memory, RAM, etc). Even if all the kernels have the same purpose, however they all have different architecture, each of them has its own features, advantages and disadvantages. So this article provides information about kernel functions and kernel architectures with their features, advantages and disadvantages.*

Keywords: *kernel, Operating System, monolithic kernels, microkernels, hybrid kernels, nanokernel, exokernel.*

Introduction

The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system. It handles the rest of start-up as well as input/output requests from software, translating them into data-processing instructions for the central processing unit. It handles memory and peripherals like keyboards, monitors, printers, and speakers.

The kernel performs its tasks, such as running processes, managing hardware devices such as the hard disk, and handling interrupts, in this protected kernel space. This separation prevents user data and kernel data from interfering with each other and causing instability and slowness, as well as preventing malfunctioning application programs from crashing the entire operating system.

The kernel's interface is a low-level abstraction layer. When a process makes requests of the kernel, it is called a system call. Kernel designs differ in how they manage these system calls and resources.[1]

1. Basic functions of kernel

As we have seen above that kernel is the heart of any Operating System, so all the vital functions should be controlled and managed by kernel itself. There are various tasks and functions of a kernel:

1. Resource allocation - The kernel's primary function is to manage hardware resources and allow other programs to run and use these resources. These resources are - CPU, Memory and I/O devices.

2. Process Management - A process defines which memory sections the application can access. The main task of a kernel is to allow the execution of programs and support them with features such as hardware abstraction.

To run an application, a kernel first set up an address space for the program, then loads the file containing the program's code into memory, then set up a stack for the program and branches to a given location inside the program, thus finally starting its execution.

3. Memory Management - The kernel has full access to the hardware's memory. It allows processes to safely access this memory as they require it. Virtual addressing helps kernel to create virtual partitions of memory in two disjointed areas, one is reserved for the kernel (kernel space) and the other for the applications (user space).

4. I/O Device Management - To perform useful functions, processes need access to the peripherals connected to the computer, which are controlled by the kernel through Device Drivers. A device driver is a computer program that enables the operating system to interact with a hardware device. It provides the operating system with information of how to control and communicate with a certain piece of hardware.

A kernel maintains a list of available devices. A device manager first performs a scan on different hardware buses, such as Peripheral Component Interconnect (PCI) or Universal Serial Bus (USB), to detect installed devices, then searches for the appropriate drivers.

5. Inter - Process Communication - Kernel provides functions for Synchronization and Communication between processes called Inter-Process Communication (IPC). There are various procedures of IPC, semaphore, shared memory, message queue, pipe (or named FIFO), etc.

6. Scheduling - In a Multitasking system, the kernel will give every program a slice of time and switch from process to process so quickly that it will appear to the user as if these processes were being executed

simultaneously. The kernel uses Scheduling Algorithms to determine which process is running next and how much time it will be given. The mechanism sets priority among the processes.

7. System Calls and Interrupt Handling - A system call is a mechanism that is used by the application program to request a service from the OS. System calls include close, open, read, wait and write functions. To access the services provided by the kernel we need to call the related kernel functions. Most kernels provide a C Library or an API, which can call the related kernel functions.

8. Security or Protection Management - The kernels also provide protection from bugs (fault control) and from malicious software.[2]

2. Microkernel

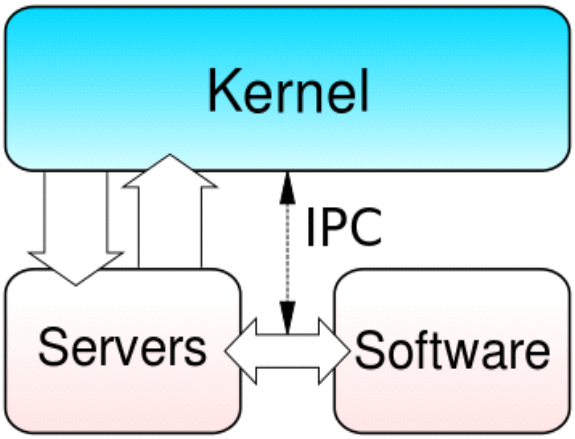

Figure 1 Structure of microkernel

The design of a microkernel determines that it has only methods for a simple IPC, memory management and scheduling. All these methods are instructions in the kernel mode but the rest runs in the user mode.

The microkernel is not implemented as one huge process. The functionality of the microkernel is separated in several processes, called Servers. In best case only these Servers get more benefits which need them to do their missions. All servers are divided from the system and each process has his own address space. The result is that the microkernel cannot start functions directly.

It has to communicate via "Message Passing" which is an IPC mechanism which allows Servers to communicate to other Servers. Because of this implementation mistakes affect only the process in which it occurs. These modularization allows to exchange servers without jamming the whole system. The communication via IPC produce more overhead then a function call and more context switches than a monolithic kernel. The result of the context switches is a major latency, which results in a negative performance.[3]

3. Monolithic kernel

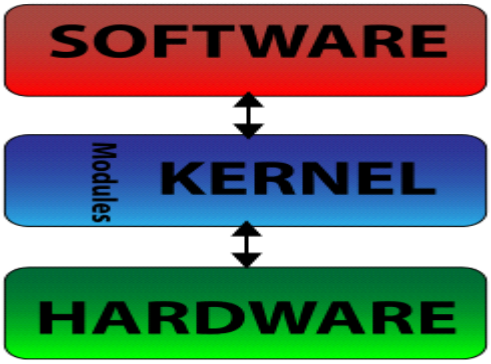

Figure 2 Structure of monolithic kernel

In contrast to the microkernel the monolithic kernel has more functions, so there are more services that run in kernel space, like all device drivers, dispatcher, scheduling, virtual memory, all IPC (not only the simple IPC as in a microkernel), the virtual file system and the system calls, so only programs can run in user-mode.

The monolithic kernel is implemented as one process, which runs in one single address space. All kernel services are running in one kernel address space, so the communication between these services is easier, because the kernel processes have the ability to call all functions directly like an application in user-space. The feature to perform system calls results in better performance and easier implementation of the kernel. The crash or bug in one module which is running in kernel-mode can crash the whole system.[3]

4. Hybrid kernel

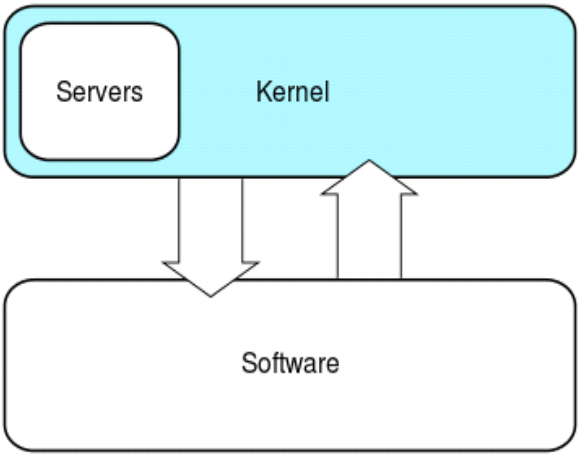

Figure 3 Structure of hybrid kernel

The hybrid kernel structure is something between the microkernel and the monolithic kernel structures, that is the reason for the name. The hybrid kernel runs the same processes in kernel-mode like a microkernel. Additional in the hybrid kernel runs the application IPC and the device drivers in kernel mode. In the user-mode is used for UNIX-Server, File-Server and user applications. The goal of this architecture is to get the performance advantages of a monolithic kernel, with the stability of a microkernel. The result is a microkernel-like structure in a monolithic kernel and a dispute about the need of an extra category for this kernel or only have the two categories: microkernel and monolithic kernel.[3]

5. Nanokernel

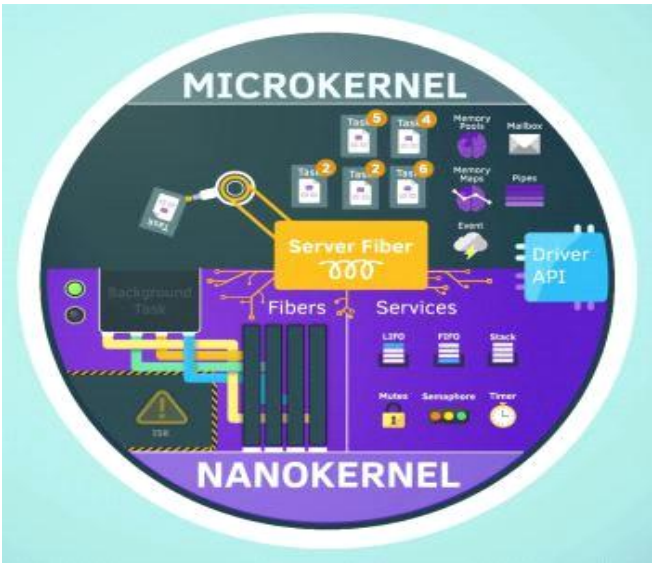

Figure 4 Structure of nanokernel

A nanokernel is a kernel that can delegate virtually all services – including even the most basic ones like interrupt controllers or the timer – to device drivers to make the kernel memory demand even smaller than a conventional microkernel.

Nanokernels are comparatively little kernels which give hardware abstraction, however they have lack system services. They provide the very basic OS functionalities and rest is implemented as applications. Such kernels are used for creating OS for very particular devices in military, science or other industries.[4]

6. Exokernel

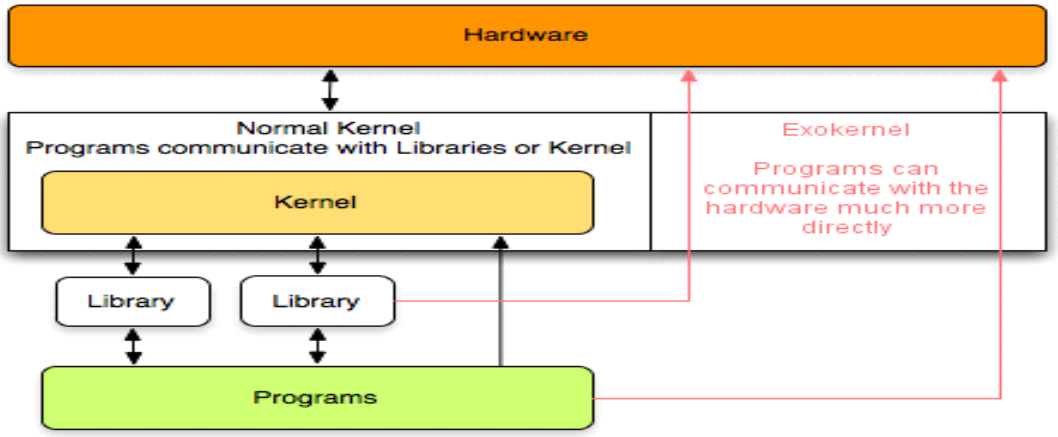

Figure 5 Structure of exokernel

Exokernel is a type of OS kernel created at the MIT that seeks to provide application-level management of hardware resources. The exokernel structure is designed to divide resource protection from management to improve application-specific customization. Exokernels have small size, because of their limited operability.

Traditional OS always have an impact on the performance, functionality and area of applications that are built on them, because the OS is positioned between the user applications and the hardware resources. The exokernel operating system attempts to address this problem by eliminating the notion that an OS must provide abstractions upon which to build programs. The idea is to impose as few abstractions as possible on the developers and to provide them with the liberty to use abstractions as and when needed. The main aim of an exokernel is to ensure that there is no forced abstraction, which is what makes an exokernel different from micro- and monolithic kernels.[5]

Some of the features of exokernel operating systems include:

- Superior support for application control
- Divides security from management
- Abstractions are moved securely to an untrusted library operating system
- Provides a low-level interface
- Library operating systems provide portability and compatibility

The benefits of the exokernel operating system include:

- Improved performance of programs
- More efficient use of hardware resources through precise resource allocation and revocation
- Easier development and testing of new operating systems
- Each user-space program is allowed to apply its own optimized memory management

Some of the drawbacks of the exokernel operating system include:

- Reduced consistency
- Complex design of exokernel interface

In conclusion, kernel is the heart of every modern Operating System, because it includes the most important modules, functions and instructions that are used by the OS.

References:

1. http://www.linfo.org/kernel.html
2. https://www.quora.com/What-are-the-functions-of-kernel
3. https://www.mi.fu-berlin.de/inf/groups/ag-tech/teaching/2008-09_WS/S_19565_Proseminar_Technische_Informatik/bitterling09operating.pdf
4. https://en.wikipedia.org/wiki/Microkernel#Nanokernel
5. https://www.techopedia.com/definition/27006/exokernel