Kernel Design in Operating System

[1]Name of 1st Author: Muthu Dayalan
[1]Designation of 1st Author: Senior Software Developer,Chennai, India

Abstract- **The kernel is a crucial component in the operating system. This paper presents the three main types of kernels used in operating systems and their benefits and drawbacks in the applications. The microkernel, monolithic kernel, and the hybrid kernels are the three major types that have been outlined and their features. Two major operating systems of desktop computers; Windows Vista and Linux have two different kernel subsystems. Their kernels, though having the same architecture have different subsystems have very different features, which the paper has tried to discuss at length. Some of the notable differences include; process scheduling, process management, memory management and synchronization of the kernel.**

Keywords: **microkernel, hybrid, monolithic kernel, Windows Vista OS, Linux OS, Kernel subsystems**

I. INTRODUCTION

Several years after the development of the operating systems, kernels came into the picture and enhanced the operation of the computers. Basically, kernels from the core of the operating system. They are responsible for providing secure access to computer hardware and run programs. Some of the common activities that kernels do include scheduling, buffering, caching, device reservation and spooling, error handling, and input/output protection Microkernels were the first kernels to be used in the original operation systems [1]. A microkernel is a minute operating kernel system of a computer. These kernels were small, initially as the computers too had limited memory and with the advancement of computers to more and efficient functions kernel designs were also improved to enable control of more number of devices. With the increase of the address spaces from 16 to 32 bits the kernel designs were also improved and were not limited to the software adventure. The monolithic kernel took over from the microkernel which is mainly with the Linux-kernel and FreeBSD (BSD-derivatives) [4]. The monolithic kernel was also used by MS-DOS as well as the MS Windows 9x series. In addition to the two designs, there are the hybrid designs which are mainly used in the series of Windows NT (XP and Vista), as wells as Mac's OS/2 and OS/X.

II. KERNEL ARCHITECTURE

The following sections discuss the design differences of the three types of kernels; microkernel, monolithic and the hybrid kernels.

Microkernels

This architecture was designed for simple operations in the computer such as protocol stack, basic driver services, and file system among others. The kernel space is therefore reduced due to the minimal operations, which in turn increases the stability and security of the operating system as the bare minimum code is running the kernel [4]. A crash of services like network service because of buffer overflow will only make the networking service corrupted but leave the other system still functional. The microkernel functionality is divided into various processes called servers. These servers have their own addresses and are all separated from the system which makes microkernel not able to start functions directly. Therefore, the microkernel has to communicate through "Message Passing", an inter-process

communication that allows servers to communicate with other servers [15]. This communication is the feature that allows implementation errors to affect the processes where it only occurs. However, since the communication uses context switches, there is a major latency, which affects the performance of the microkernel negatively.

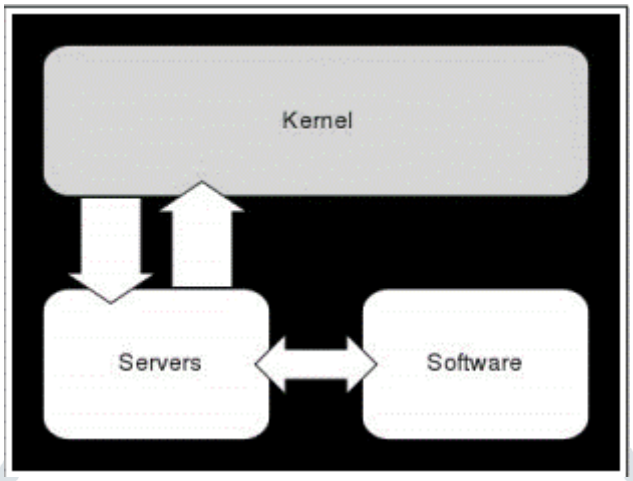

Figure 1: A visual representation of microkernel [1]

Monolithic Kernel

Unlike the microkernel, the monolithic kernel is able to run more services due to the extra number of functions. The implementation process of this type of kernel is in one process and runs on a single address. Therefore, communication between various services in the computer is more simplified as the kernel processes have the ability to call the functions directly as a program would do in a user-space [14]. This ability to perform the system call leads to better performance of the kernel and simpler implementation. However, the monolithic kernel has a tendency of generating more bugs and errors since the same address locations are used by the same address processes [15]. Additional, it is more tedious to add or remove monolithic features as it requires to rewrite the whole kernel.

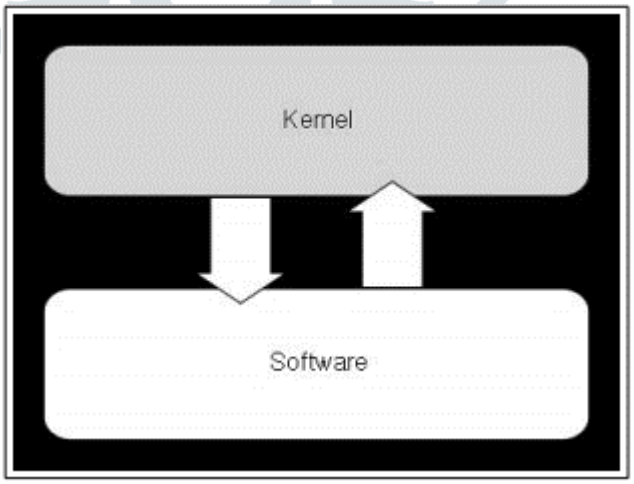

Figure 2: A visual representation of microkernel [1]

Hybrid Kernel

The hybrid kernel is a synergy of both the microkernel and monolithic kernel. It runs a similar process as microkernel but also runs the device drivers and the application IPC in kernel mode [15]. The aim of this architecture is to make a structure that has benefits of the monolithic kernel but has the stability of microkernel.

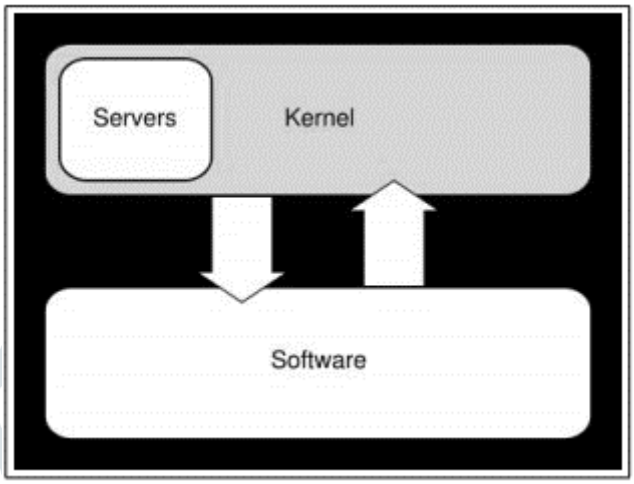

Figure 3: A visual representation of microkernel [1]

III. KERNEL SUBSYSTEMS

The following section discusses the kernel subsystems of Linux and Windows Vista.

Linux Kernel Architecture

Linux has a monolithic kernel which is also modular. The Linux kernel is useless in isolation, as it makes a crucial part of the larger system, thus sensible to discuss it in the context of the connected system. The following is the composition of the Linux operating system:

a) O/S Services – These are the services that are taken as part of the operating system [2]. They include command shell, windowing system among others.
b) User Applications – These refer to sets of applications that are used in a particular Linux system and are different depending on the use of the computer. Examples of these include; word processor and web browser.
c) Linux Kernel – the kernels act as a bridge to access the hardware resources and the central processing unit (CPU) [8].
d) Hardware Controllers – the subsystem comprises of the physical devices found in a Linux installation such as the CPU, hard disks, memory hardware, and network hardware [2].

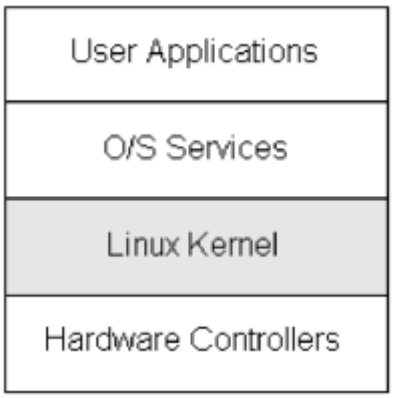

Figure 4: The composition of a Linux operating system

The purpose of the Linux kernel is to present the user process virtual machine interphase. Usually, the process is coded without the need to know the physical hardware that is stored in a computer. Therefore, the Linux kernel abstracts all the hardware that is composed in the computer to act consistently with the virtual interface. It also supports multitasking of various processes as well as mediating access to hardware resources.

The composition of the Kernel Structure

i) Process Management in Linux Kernel: Information in the Linux kernel is saved in a data structure, which contains varies information like process status, process address space, and open file. The processes are identified by a process identifier (PID). The processes in Linux are usually in five states, RUNNING (the task is running), INTERRUPTIBLE (the process is blocked and awaits a signal), UNINTERRUPTIBLE (the process cannot wake), ZOMBIE (the process is complete but the parent has made a wait call to the system) and STOPPED (the process is stopped and cannot wake) [1]. To create a new process in Linux kernel, function fork () and exec () are used to create a copy of the actual file of the tack

ii) Process Scheduler (SCHED): it controls the process accesses to the CPU by enforcing a policy that ensures there is fair access to the CPU by all processes. It also ensures that the actions of the hardware are performed in time [1]. The system uses a Completely Fair Scheduler (CFS)to schedule the processes. The tasked are tracked and balanced using per-task p->wait_runtime expressed in nanoseconds to fairly balance the processes running in the system [11].

iii) Virtual File System (VFS): The VFS presents a common file interface to all the devices so as to abstract their details, in addition to supporting several file formats

iv) Memory Manager (MM): It permits the use of the main memory system of the machine securely by multiple processes. It also supports the virtual manager that supports the processes that use more memory than what is available. The memory is usually organized in blocks refresh to like pages and are of different sizes [1]. The system uses three different page zones: a) ZONE_DMA for pages that are DMA-enabled, b) ZONE_NORMAL for pages that are regular and c) ZONE_HIGHMEM for pages that do not have permanent address space in the kernel [3]. The operations inside the kernel are responsible for allocating and freeing the data structures.

v) Network Interphase (NET): It provides the system to various standards of network and also networks hardware [8].

vi) Inter-Process Communication (IPC): This subsystem supports various mechanisms used for process-to-process communication in Linux system.

IV. WINDOWS KERNEL ARCHITECTURE

The Windows operating system uses a hybrid kernel subsystem. A monolithic kernel as emulation system is run in the user mode in this system with the structure having a collection of modules that communicate through interfaces [14]. The systems then encompass a small microkernel which has its operations limited to just core functions such as thread scheduling, first-level interrupt handling, and synchronization primitives. This architecture allows the possibility of having direct calls procedure or inter-process communication for communication between modules, and thus for the potential modules location in varied address spaces.

Process Management

In Vista, the unit for a thread is referred to as a thread. Each container in Vista contains at least one thread which that ability to start a process and create additional threads. A process in Vista has a unique process identifier, priority class, executable code, and can handle the system objects [5]. Additionally, each thread in the system contains a scheduling priority, a unique process identifier as well as structures that the system uses to save the thread until the thread has been scheduled.

Process Scheduling

Since the processes contain one or more threads in Vista, the threads have to be scheduled. The system supports preemptive multitasking, therefore can execute many threads simultaneously in relation to the number of processors in the computer [1]. A group of processes are managed as a unit in Vista and are referred to as a job object. All the operations that are performed on the job object affect the processes that are associated with the job object. All processes in Vista get a slice time to run the threads. A priority level dictates the time each thread will learn [5]. The priority level of each thread is determined by the priority class of its process. The scheduling of the threads involves the threads with high priority level the first opportunity to run. If a low priority thread is running, and a high priority thread comes up, the process scheduler terminates the running of the low priority thread and starts the high priority thread.

Memory Management

Each process has a virtual address space which at times can be larger than the available physical memory of the computer. The virtual address set that resides in the physical memory of the computer is offered to as a working set. In a situation where the threads of a particular process use the physical memory bigger than what is currently available some memory contents are paged into the disk by the system [12]. Therefore, the total amount of the virtual address space that is available on the system is not pegged to the physical size of the memory. The process of paging involves moving the recently unused physical memory page to the paging file [13]. Due to the limitations of the RAM (random access memory), the process on Windows Vista is likely to share a page until one process needs to write on the page.

Kernel Synchronization

Objects in Windows Vista needs to be synchronized to access the shared data due to the multitasking nature of the operating system. The kernel creates, maintains and signals the synchronization of objects [12]. A process waiting for that synchronization object is stated as waiting by the dispatcher and then to ready when it has been unblocked to run. Spinlocks are used to loop the processes when they are unlocked. Kernel transaction manager (KTM), similar to Linux atomic operations enables the use of atomic applications by availing them as kernel objects [13].

V. CONCLUSION

The paper explains at large the usage of the monolithic and hybrid kernel architecture in the current system of Linux and windows vista. Linux which is in fast development is using the monolithic kernel to run its operating system which is mainly compatible with its architecture. The Windows Vista has adopted the hybrid kernel to enjoy the benefits of monolithic as well as the security and stability of the microkernel. Though the Linux and windows vista performs almost similar systems to a user, their architecture and configuration show unique differences as to how the processes and tasks are performed.

REFERENCES

[1] Bitterling, P. Operating System Kernels.

[2] Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel: from I/O ports to process management*. " O'Reilly Media, Inc.

[3] Chapter 12. Memory Management. (n.d.). Retrieved from https://notes.shichao.io/lkd/ch12/

[4] Cheriton, D. R., & Duda, K. J. (1994, November). A caching model of operating system kernel functionality. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation* (p. 14). USENIX Association.

[5] Gite, V. (2007, April 23). Comparison: Windows XP / Vista kernel vs Linux Kernel. Retrieved from https://www.cyberciti.biz/tips/windows-xp-vista-kernel-vs-linux-kernel.html

[6] Gleixner, T. (2008). "The completely fair scheduler," http://www. linux-foundation.jp/uploads/seminar20080709/lfjp2008.pdf,

[7] Hu, Y., Kwon, Y., Chidambaram, V., & Witchel, E. (2017, May). From Crash Consistency to Transactions. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (pp. 100-105). ACM.

[8] Love, R. (2005) Linux-Kernel-Handbuch. ADDISON-WESLEY,

[9] Love, R. (2010). *Linux Kernel Development: Linux Kernel Development _p3*. Pearson Education.

[10] Mauerer, W. (2010). *Professional Linux kernel architecture*. John Wiley & Sons.

[11] Pabla, C. S. (2009). Completely Fair Scheduler. *Linux Journal*. Retrieved from https://www.linuxjournal.com/node/10267

[12] Probert, D. (2008). *The architecture of the Windows Kernel* [pdf]. Retrieved from https://www.cs.fsu.edu/~zwang/files/cop4610/Fall2016/windows.pdf

[13] Russinovich, M. (2007). Inside the windows vista kernel: Part 3. *Microsoft TechNet Magazine*.

[14] Stallings, W. (2012). *Operating systems: internals and design principles*. Boston: Prentice Hall.

[15] What is Operating System, Kernel and Types of kernels. (n.d.). Retrieved from http://www.go4expert.com/articles/operating-kernel-types-kernels-t24793/