**OPERATING SYSTEM ARCHITECTURE**

-Sai Sirisha Nadiminti (2020201044, PG1)

*“It is better to debate a question without settling It than to settle a question without debating it”.*

*- Joseph Jubert*

“LINUX is obsolete”. This line posted by Adrew S. Tanenbaum to “comp.os.minix” (a public newsgroup) on 29 Jan 1992 started the famous Linus-vs-Tanenbaum debate over the suitable architecture of Operating System. Deciding the architecture of the operating system is vital as it will determine if the OS can function properly and be modified easily. Let us look at various types of architecture[[1]](#endnote-26577) before going into the details of the debate.

**MONOLITHIC ARCHITECTURE**

It is a single, static, binary file that runs in a single address space. The kernel provides the file system, CPU scheduling, memory management and other operating system functions through system calls.[[2]](#endnote-15840)

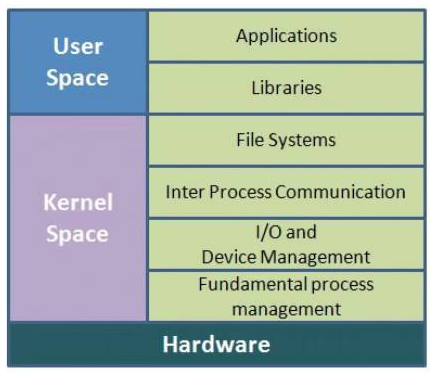


Fig 1. Monolithic kernel

**LAYERED ARCHITECTURE**

In layered approach, the operating system is broken into several levels. The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface. A typical OS layer consists of data structures and a set of routines that can be invoked by higher-level layers. The layer can itself invoke operations on lower-level layers.

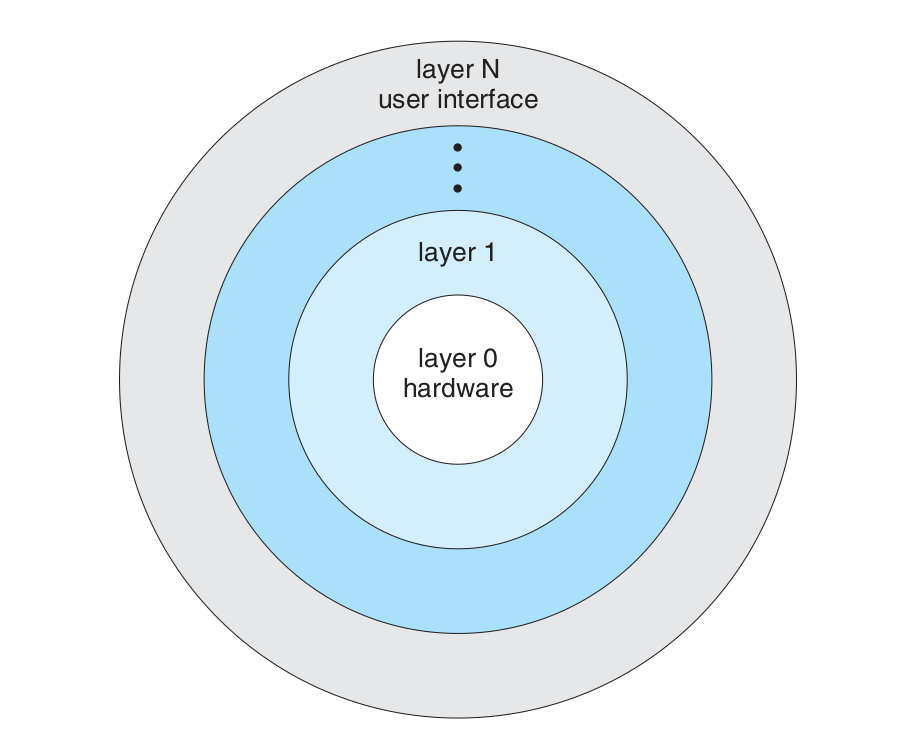


Fig 2. A layered operating system

**MODULAR ARCHITECTURE**

In the modular architecture, the kernel has a set of core components and links in additional services via modules, either at boot time or during run time. The idea of the design is for the kernel to provide core services while other services are implemented dynamically, as the kernel is running. For example, we might build CPU scheduling and memory management algorithms directly into the kernel and then add support for different file systems by way of loadable modules.

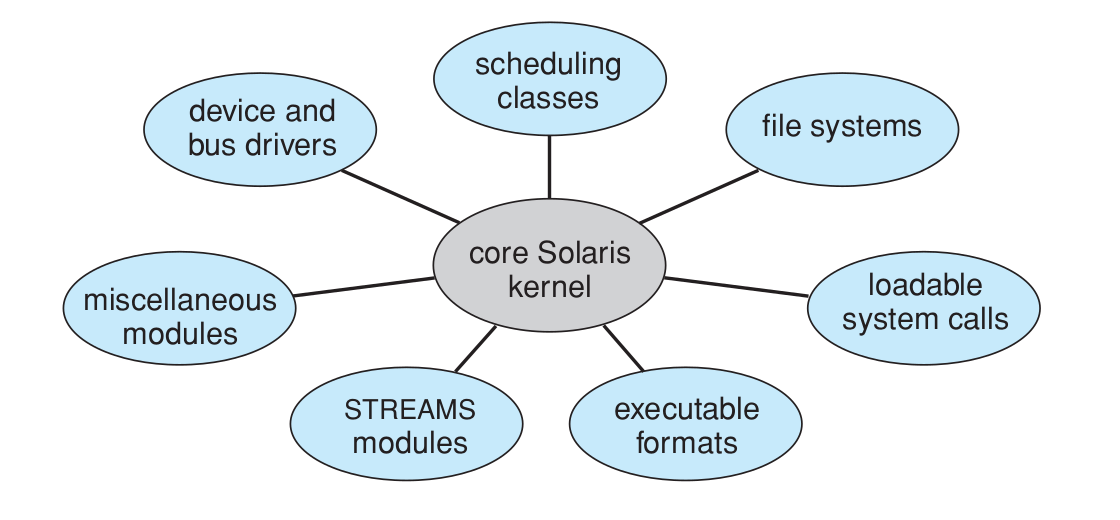


Fig 3. Solaris loadable modules

**PART A: Advantages to having a layered/modular/monolithic architecture for OS:**

|  |  |
| --- | --- |
| **ARCHITECTURE** | **ADVANTAGES** |
| Monolithic Architecture | * This type of operating system has a simple structure. All the components needed for processing are embedded into the kernel. * All the components can directly communicate with each other and with the kernel. There is very little overhead in the system call interface or in the communication with the kernel. So, it gives high speed and efficiency. * The code to make monolithic kernel is very fast and robust. |
| Layered Architecture | * Implementers have more freedom in changing the inner workings of the system. * Information hiding leaves programmers free to implement the low-level routines as they see fit, provided that the external interface of the routine stays unchanged and that the routine itself performs the advertised task. * Simplicity of construction, debugging and system verification. |
| Modular architecture | * Each kernel section has defined, protected interfaces. It is more flexible than a layered system, because any module can call any other module. * The approach is also like the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules; but it is more efficient, because modules do not need to invoke message passing in order to communicate. Linux also uses loadable kernel modules, primarily for supporting device drivers and file systems. |

I read the entire debate over the architecture of LINUX that happened over emails. It was amusing to read the witty jibes that they shot for each other.

**PART B: Key points from the Linus-vs-Tanenbaum debate on monolithic kernels:**

The gist of the debate was: Tanenbaum suggested to make LINUX micro-kernel based and Linus wanted to stick to the basic monolithic design to improve performance as the microkernels use message-passing that decreases the efficiency of the OS.[[3]](#endnote-23846)

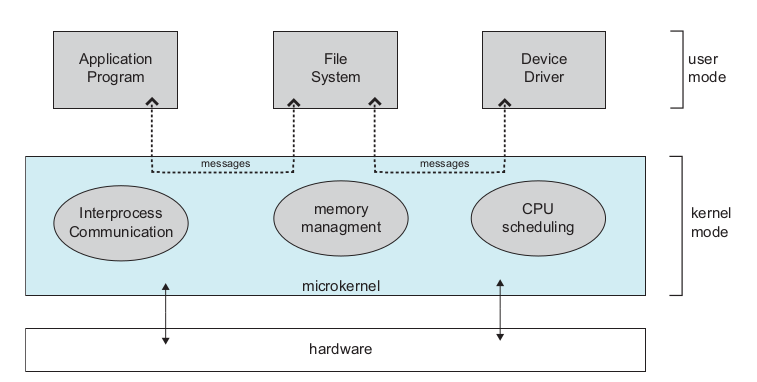


Fig 4. Architecture of a typical microkernel

1. Tanenbaum wrote on 29 Jan 1992. He argued in favor of microkernels over monolithic architecture. He said that making LINUX architecture monolithic would be taking a giant step backwards in 1970s and it would be like rewriting an existing working C code in BASIC. Tanenbaum also mentioned that LINUX is closely tied with 80x86 microprocessor. During that time, hardware was emerging at a very fast pace. Tanenbaum mentioned that using a microkernel-based portable OS will be a wise choice.

*“Portability is for people who cannot write new programs” - Linus Torvalds*

1. To Tanenbaum’s message, Linus replied that the architecture of the operating system should not be the only criteria for deciding about the “goodness” of the OS. Taking about portability, Linus said that he had made a conscious choice of trading portability for simpler design and it does not make any sense to make any OS overtly portable.

*“I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-). Writing a new OS only for the 386 in 1991 gets you your second 'F' for this term. But if you do real well on the final exam, you can still pass the course.” - Andrew Tanenbaum*

1. Tanenbaum replied to this jibe by saying that it is funny that Linus made LINUX free but expects people to buy an expensive hardware to run LINUX on. He reiterated that writing an operating system for a particular piece of hardware is not wise.

*“That's ok. Einstein got lousy grades in math and physics.” - Linus Torvalds*

1. Linus replied that even though MINUX ends up being portable to even the machine that does not suit the hardware of MINUX. So, all the features like paging end up getting wasted anyway. Making the LINUX monolithic saves ugly message passing and makes it inherently multithreaded as every process does its own job. About portability, Linus mentioned that LINUX API will be portable as it is built on UNIX.

What I personally found amazing is that Linus did not have any future long-term vision for LINUX. It was just one of his pet-projects. Everyone who used LINUX majorly installed it because it was free and suited to their needs at that time, basically put, it suited their short-term goals well. They could look at the code, easily understand it and install it for free and toy with it (maximum functionality in the minimum amount of time/cost/hassle). Nobody needed portability as they had to operate on only a single system.

Many years later, Linus was asked that companies like Google and Android make a fortune out of LINUX, that itself is free and how does Linus feel about it. Linus gave a beautiful reply that the reason why LINUX became so big was because it was open-source, free and simple to understand. So, many people used it, supported it and enhanced it. LINUX allowed itself to become expanded, unlike other operating systems that were closed to such improvements. Even Linus himself did not expect LINUX to blow up like this. He said himself that LINUX is tied to 80386. In retrospect, Andrew Tanenbaum saying “Linux is obsolete” in 1992 sounds ironical. Moving forward, we all know that the spread of internet just helped LINUX even more. It was not so obsolete after all.

*“Your approach to people requesting features in Minix, has generally been to tell them that they didn't really want that feature anyway.” - Douglas Graham*

From the debate, I learnt that it is important to stick to simple designs that work beautifully. Adding features that are just added for the sake of it without giving any performance gain makes no sense and come at a performance cost (like message passing and system overheads in case of microkernels).

LINUX was just 12000 lines of codes. Converting it into microkernel and adding the overhead of message passing did not make sense then.

*“There are two kinds of researchers: those that have implemented something and those that have not. The latter will tell you that there are 142 ways of doing things and that there isn't consensus on which is best. The former will simply tell you that 141 of them don't work." - David Cheriton (Prof. at Stanford, and author of the V system)*

Even though microkernels seemed a good idea in practice, they degraded the performance of the system. Many researches supported microkernel only as a great learning tool. But many of them preferred LINUX in their personal computers. Many “hackers” came on to support LINUX for its open source (It still is the best free operating system), the efficiency with which it operates on 80x86 and that it gives a programmer a tool to learn. What started as a self-project in Linus Torvald’s den has now become the transformative system on which modern android phones are being built today.

**PART C: Linux overcomes the disadvantages of a monolithic kernel by being modular monolithic*.***

*Perhaps the features that people want for either functionality or compatibility could be offered by run-time loadable modules/libraries that offer these features. The micro-kernel would still be a base-level resource manager that also routes function requests to the appropriate module/library. The modules could be threads or user processes.* - David Miller

Monolithic: All functionality is compiled into the same static binary that is loaded into memory on boot. Modular: In the beginning, only the core of the kernel is loaded. Other modules are loaded as per need and unloaded when they are not needed.

All the advantages of having a monolithic kernel persist in LINUX architecture. Having the operating system in a single address space provides very efficient performance. As all kernel code and data structures are kept in a single address space, no context switches are necessary when a process calls an operating-system function or when a hardware interrupt is delivered. Moreover, the kernel can pass data and make requests between various subsystems using relatively cheap C function invocation and not more complicated inter-process communication (IPC). This single address space contains not only the core scheduling and virtual memory code but all kernel code, including all device drivers, file systems, and networking code.

Limitations of a monolithic kernel are:

* Code written in this operating system (OS) is difficult to port.
* Monolithic OS has more tendency to generate errors and bugs. The reason is that user processes use same address locations as the kernel.
* Adding and removing features from monolithic OS is very difficult. All the code needs to be rewritten and recompiled to add or remove any feature.

These limitations are overcome by LINUX as we have made the OS modular now. The kernel can load and unload independent modules dynamically at run time on demand. These loadable kernel modules run in privileged kernel mode and therefore have full access to all the hardware capabilities of the machine on which they run. Kernel modules allow a Linux system to be set up with a standard minimal kernel, without any extra device drivers built in. Any device drivers that the user needs can be either loaded explicitly by the system at startup or loaded automatically by the system on demand and unloaded when not in use. For example, a mouse driver can be loaded when a USB mouse is plugged into the system and unloaded when the mouse is unplugged.

Since the modules are independent of each other, different features can be added or removed easily. Modular components can be reused and repurposed. This can lead to faster and more consistent development. Because modular monoliths make individual functions more independent, they, by nature, make dependencies more organized and visible. This makes it easier for developers to easily assess which parts of the application require which dependencies. This makes the debugging easier as we just can now focus on the module that has caused an issue rather than the entire kernel.

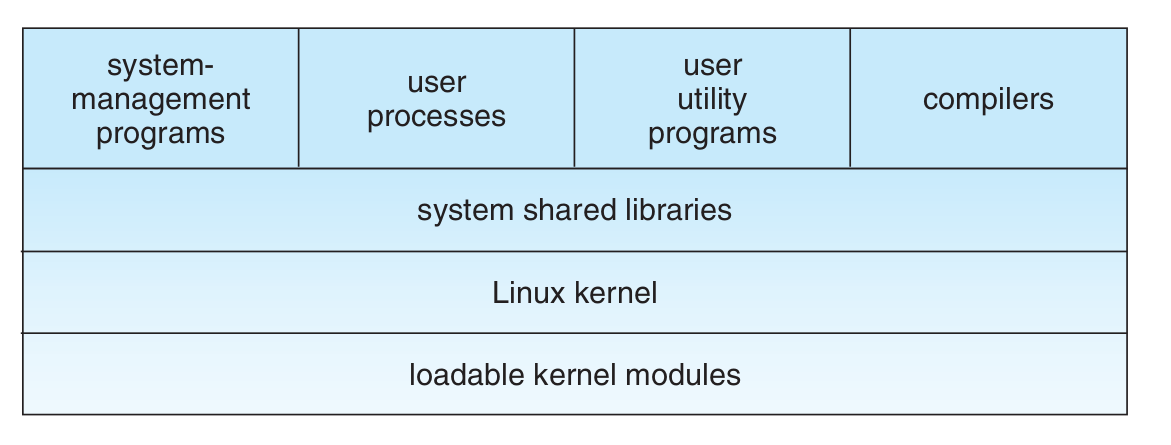


Fig 5. Linux System

This is how LINUX beautifully merges the advantages of the monolithic as well as the modular architecture.[[4]](#endnote-25612)

1. Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

   Note: All the information about the architectures is taken from this book. [↑](#endnote-ref-26577)
2. https://www.itrelease.com/2018/07/what-is-a-monolithic-operating-system/ [↑](#endnote-ref-15840)
3. <https://www.oreilly.com/openbook/opensources/book/appa.html> [↑](#endnote-ref-23846)
4. <https://www.jrebel.com/blog/what-is-a-modular-monolith> [↑](#endnote-ref-25612)