Why is Linux called a monolithic kernel?

I read that Linux is a monolithic kernel. Does monolithic kernel mean compiling and linking the complete kernel code into an executable?

If Linux is able to support modules, why not break all the subsystems into modules and load them when necessary? In that case, the kernel doesn't have to load all modules initially and could maintain an index of the functions in the module and load them when necessary.

architecture operating-system linux-kernel

edited Sep 20 '11 at 6:19
Peter Mortensen
7,802 8 55 90

asked Nov 27 '09 at 3:16
Boolean
3,412 13 57 97

6 Answers

A monolithic kernel is a kernel where all services (file system, VFS, device drivers, etc) as well as core functionality (scheduling, memory allocation, etc.) are a tight knit group sharing the same space. This directly opposes a *microkernel*.

A microkernel prefers an approach where core functionality is isolated from system services and device drivers (which are basically just system services). For instance, VFS (virtual file system) and block device file systems (i.e. minixfs) are separate processes that run outside of the kernel's space, using IPC to communicate with the kernel, other services and user processes. In short, if it's a *module* in Linux, it's a *service* in a microkernel, indicating an isolated process.

Do not confuse the term *modular* kernel to be anything but monolithic. Some monolithic kernels can be compiled to be modular (e.g Linux), what matters is that the module is inserted to and runs from the same space that handles core functionality.

The advantage to a microkernel is that any failed service can be easily restarted, for instance, there is no kernel halt if the root file system throws an abort.

The disadvantage to a microkernel is that asynchronous IPC messaging can become very difficult to debug, especially if fibrils are implemented. Additionally, just tracking down a FS/write issue means examining the user space process, the block device service, VFS service, file system service and (possibly) the PCI service. If you get a blank on that, its time to look at the IPC service. This is often easier in a monolithic kernel. GNU Hurd suffers from these debugging problems (reference). I'm not even going to go into checkpointing when dealing with complex message queues. Microkernels are not for the faint of heart.

The shortest path to a working, stable kernel is the monolithic approach. Either approach can offer a POSIX interface, where the design of the kernel becomes of little interest to someone simply wanting to write code to run on any given design.

I use Linux (monolithic) in production. However, most of my learning, hacking or tinkering with kernel development goes into a microkernel, specifically HelenOS.

Edit

If you got this far through my very long-winded answer, you will probably have some fun reading the 'Great Torvalds-Tanenbaum debate on kernel design'. It's even funnier to read in 2013, more than 20 years after it transpired. The funniest part was Linus' signature in one of the last messages:

```
Linus "my first, and hopefully last flamefest" Torvalds
```

Obviously, that did not come true any more than Tanenbaum's prediction that x86 would soon be obsolete.

NB:

When I say "Minix", I do not imply Minix 3. Additionally, when I mention The HURD, I am referencing (mostly) the Mach microkernel. It is not my intent to disparage the recent work of others.

edited May 21 '13 at 19:17

community wiki
18 revs, 5 users 79%
Tim Post

3 Interestingly Linus Torvalds was greatly influenced by Andew Tanenbaum's MINIX when he created Linux. However, MINIX is based on a micro kernel design while Linux uses a monolithic kernel. – Martin Liversage Nov 27 '09 at 8:12

1 @Martin Liversage: More frustrated than influenced :) I edited my answer to reflect that. – Tim Post ♦ Nov 27 '09 at 12:28

2 Hehe, yes, LT developed into one of the net's most consistent flamers... – DigitalRoss Nov 28 '09 at 0:05

14 @DigitalRoss: You should see my inbox after answering this, Linus is tame compared to Minix and Mach enthusiasts. – Tim Post ♦ Nov 29 '09 at 17:05

1 Tangentially related: Why was Tanenbaum wrong in the Tanenbaum-Torvalds debates? – Yannis Oct 22 '12 at 15:03

Monolithic kernel means that the whole operating system runs in kernel mode (i.e. highly privileged by the hardware). That is, no part of the OS runs in user mode (lower privilege). Only applications on top of the OS run in user mode.

In non-monolithic kernel operating systems, such as Windows, a large part of the OS itself runs in user mode.

In either case, the OS can be highly modular.

answered Nov 27 '09 at 3:19


CesarGon
8,987 3 27 63

5 Windows is most definitely a monolithic kernel. – Adam Rosenfield Nov 27 '09 at 3:51

3 @Adam: I disagree. The old-style 16-bit Windows *was* monolithic kernel, as was Windows 95 and the like. But NT-based editions of Windows, including all Server versions plus Vista and 7, are clearly microkernel or perhaps hybrid, depending on what definition of "microkernel" you use. – CesarGon Nov 27 '09 at 3:59

5 Just because the printer drivers don't run in ring0 doesn't make it a microkernel :) – caf Nov 27 '09 at 5:40

3 @caf: I suggest you take a look at en.wikipedia.org/wiki/Windows_NT_kernel and en.wikipedia.org/wiki/Comparison_of_operating_system_kernels. You'll see that Windows NT and their successors, including Vista, 7 and the Servers, are described as "hybrid kernel". Two large subsystems of the OS run fully in user mode, not just a printer driver. :-) – CesarGon Nov 27 '09 at 15:23

3 My comment was somewhat tongue-in-cheek - the "hybrid" designation seems so information-free as to be useless. – caf Nov 28 '09 at 1:31

From wikipedia

> A monolithic kernel is a kernel architecture where the entire operating system is working in the kernel space and alone as supervisor mode. In difference with other architectures,[1] the monolithic kernel defines alone a high-level virtual interface over computer hardware, with a set of primitives or system calls to implement all operating system services such as process management, concurrency, and memory management itself and one or more device drivers as modules.

Recent versions of Windows on the other hand use a Hybric kernel.

> A hybrid kernel is a kernel architecture based on combining aspects of microkernel and monolithic kernel architectures used in computer operating systems. The category is controversial due to the similarity to monolithic kernel; the term has been dismissed by some as simple marketing. The traditional kernel categories are monolithic kernels and microkernels (with nanokernels and exokernels seen as more extreme versions of microkernels).

edited Jan 28 '13 at 15:40
Reno
23.3k 6 50 73

answered Nov 27 '09 at 3:19
Bob
38.6k 17 89 103

6 If I ever do anything in kernel space, I have to remember to use "hybric kerkel" somewhere. SCNR ;-) – Jürgen A. Erhard Dec 26 '09 at 19:15

Windows NT was *always* a hybrid system. The kernel might be not exactly hybrid, but you run into question of what you count as part of it (winapi, for example, is implemented as user-space service) – p_l Jan 9 '13 at 12:12

'Monolithic' in this context does not refer to there being a single large executable, and as you say, there Linux supports the dynamic loading of kernel modules at runtime. When talking about kernels, 'monolithic' means that the entire operating system runs in 'privileged' or 'supervisor' mode, as opposed to other types of operating systems that use a type of kernel such as a 'microkernel', where only a minimal set of functionality runs in privileged mode, and most of the operating system runs in user space.

Proponents of microkernels say that this is better because smaller code means less bugs, and bugs running in supervisor mode can cause much greater problems than in user space code (such as a greater chance of having security vulnerabilities or total system crashes in the form of a 'kernel panic'). Some microkernels are sufficiently minimal that they can be 'formally verified', which means you can mathematically prove that the kernel is 'correct' according to a specification. L4 is a good example of this.

edited Sep 20 '11 at 6:22
Peter Mortensen
7,802 8 55 90

answered Nov 27 '09 at 3:20
David Claridge
2,426 1 11 21

Check your sources. The wikipage is uncited. www2.cs.uh.edu/~rzheng/course/COSC6397sp2008/... – monksy Nov 27 '09 at 3:29

Monolithic kernel is a single large processes running entirely in a single address space. It is a single static binary file. All kernel services exist and execute in kernel address space. The kernel can invoke functions directly. The examples of monolithic kernel based OSs are Linux, Unix.

I think this post will help you more to understand the concept.

http://learnlinuxconcepts.blogspot.in/2014/03/what-are-monolithic-and-micro-kernels.html

edited Mar 9 '14 at 14:38

answered Mar 9 '14 at 11:39
user3287223
24 5

While this link may answer the question, it is better to include the essential parts of the answer here and provide the link for reference. Link-only answers can become invalid if the linked page changes. – Scott Mar 9 '14 at 12:04

> ;tl-dr - No, Linux is always monolithic.

Linux *modules* may mean *modular* in some sense. As others have noted monolithic is usually representing a *microkernel* versus *monolithic* kernel. A traditional *microkernel* only has these features,

1. Scheduling
2. Memory management

3. Inter-process communications

There are no *hardware drivers*, *protocol stacks*, *filesystems*, *suspend/resume*, *clock management*, etc in the main kernel. These things are *identical* to any user task (although they may have different privileges via the MMU/scheduler).

Tanenbaum's predictions

1. Microkernels are the future
2. x86 will die out and RISC architectures will dominate the market
3. (5 years from then) everyone will be running a free GNU OS

PC and server programmers may laugh, but two and three are certainly true for the majority of cell phones in existence. Tanenbaum would be right on all accounts if BlackBerry QNX was a success.

Also, many L1-hypervisors have a micro-kernel underneath. This is because a hyper-visor usually doesn't do much beside *context* switch.

Apparently three predicts the success of Linux. ;-)

An argument for *microkernels* is that all of the monolithic sub-systems need to synchronize multiple values at one time. In order to do this, they must use locks and will suffer from Amdahl's law when extended to parallel architectures. The counter is that *microkernels* result in lots of IPC messages.

A major development is the use of lock-free programming to avoid contention in a monolithic kernel. This avoids the locking in a monolithic kernel while also reducing IPC overhead. Recently all CPUs have been extending their ISA to include better primitives for *lock-free* algorithms. So Linux will probably remain a monolithic kernel for some time.

answered Oct 27 '14 at 20:32

community wiki
artless noise

Yes, I know Tanenbaum meant Hurd. But GNU switched to Linux so the wording is funny. – artless noise Oct 27 '14 at 21:06