Linux Kernal II

9.1 Architecture: The **Linux kernel** is a Unix-like operating system kernel used by a variety of operating systems based on it, which are usually in the form of Linux distributions. The Linux kernel is a prominent example of free and open source software.

Programming language

The Linux kernel is written in the version of the C programming language supported by GCC (which has introduced a number of extensions and changes to standard C), together with a number of short sections of code written in the assembly language (in GCC's "AT&T-style" syntax) of the target architecture. Because of the extensions to C it supports, GCC was for a long time the only compiler capable of correctly building the Linux kernel.

Compiler compatibility

GCC is the default compiler for the Linux kernel source. In 2004, Intel claimed to have modified the kernel so that its C compiler also was capable of compiling it. There was another such reported success in 2009 with a modified 2.6.22 version of the kernel.

Since 2010, effort has been underway to build the Linux kernel with Clang, an alternative compiler for the C language; as of 12 April 2014, the official kernel could almost be compiled by Clang. The project dedicated to this effort is named *LLVMLinxu* after the LLVM compiler infrastructure upon which Clang is built. LLVMLinux does not aim to fork either the Linux kernel or the LLVM, therefore it is a meta-project composed of patches that are eventually submitted to the upstream projects. By enabling the Linux kernel to be compiled by Clang, which, among other advantages, is known for its fast compilation times against GCC, the kernel developers may benefit from a faster workflow due to shorter compilation times.

Interfaces

Four interfaces are distinguished: two internal to the kernel, and two between the kernel and userspace. Proprietary Linux graphic drivers like, e.g. the libGL-fglrx-glx bring their own proprietary binary blobs instead of utilizing the existent DRI/DRM. Because of the lack of a stable in-kernel ABI, this binary blobs have to

be adapted to each and every change by the hardware developers themselves. On the GDC 2014, AMD was exploring a strategy change.[54]

Conformance to standards is a general policy for the Linux kernel's internals.

Kernel-to-userspace API

Source code portability ensures that a C program written by conforming to a standard can be successfully compiled and run on any system that also conforms to the same standard. The relevant standards, aiming to achieve source code portability of programs, that the development of the Linux kernel, the GNU C Library, and associated utilities tries to adhere to, are POSIX and the Single UNIX Specification. However, as of February 2014, no Linux distributions are branded as "UNIX" by The Open Group, mainly because of the costs of the conformance testing.

The Linux kernel API of the Linux kernel, representing the kernel's system call interface, is composed of the available system calls.

Kernel-to-userspace ABI

Binary portability shall guarantee that any program once compiled for a given hardware platform, can be run in its compiled form on any other hardware platform that conforms to the standard. Binary portability is an essential requirement for the commercial viability of independent software vendor (ISV) applications built for the operating systems based on the Linux kernel. Binary compatibility is much more demanding than source code portability; as of February 2014, the only standard concerning itself with binary compatibility is the Linux Standard Base (LSB).

In-kernel API

There are a couple of kernel internal APIs utilized between the different subsystems and subsystems of subsystems. Some of them have been kept stable over several releases, other have not. There are no guarantees regarding the in-kernel APIs. Maintainers and contributors are free to augment or change them at any time.

Examples of in-kernel APIs include software frameworks/APIs for the following classes of device drivers:

- Video4Linux – for video capture hardware
- Advanced Linux Sound Architecture – for sound cards
- New API – for network interface controllers
- Direct Rendering Manager – for graphics accelerators
- KMS driver – for display controllers
- mac80211 – for wireless network interface controllers[55]
- WEXT – for wireless network interface controllers (obsolete).

In-kernel ABI

Strong interests in defining and maintaining a stable in-kernel ABI over several releases have been voiced repeatedly over time by parties, e.g. hardware manufactures, writing proprietary kernel modules and distributing binary-only software, e.g. device drivers.

By explicit choice the Linux kernel does not maintain a stable in-kernel ABI. The reasons are purely technical and not philosophical. Due to the absence of an in-kernel ABI that has to be kept stable over releases, the Linux kernel can keep a much higher rate of development.

Technical features[

Preemption

The Linux kernel provides preemptive scheduling under certain conditions. Until kernel version 2.4, only user processes were preemptive, i.e., in addition to time quantum expiration, an execution of current process in user mode would be interrupted if higher dynamic priority processes entered TASK_RUNNING state. Toward 2.6.x series of the Linux kernel, an ability to interrupt a task executing kernel code was added, although with that not all sections of the kernel code can be preempted.

The Linux kernel contains different scheduler classes. By default the kernel uses a scheduler mechanism called the Completely Fair Scheduler introduced in the 2.6.23 version of the kernel. Internally this default-scheduler class is also known as SCHED_OTHER, but the kernel also contains two real-time scheduling classes named SCHED_FIFO (realtime first-in-first-out) and SCHED_RR (realtime round-robin) both of which take precedence over the default class.

Through the use of the real-time Linux patch PREEMPT_RT, support for full preemption of critical sections, interrupt handlers, and "interrupt disable" code

sequences can be supported. Partial mainline integration of real-time Linux already brings some functionality to Linux mainline. Preemption improves latency, increases responsiveness, and makes Linux more suitable for desktop and real-time applications. Older versions of the kernel had a so-called big kernel lock for synchronization across the entire kernel. This was finally removed by Arnd Bergmann in 2011.

Additional scheduling policy known as SCHED_DEADLINE, implementing the earliest deadline first algorithm (EDF), was added to the Linux scheduler in version 3.14 of the Linux kernel mainline, released on 30 March 2014.

Portability

While not originally designed to be portable, Linux is now one of the most widely ported operating system kernels, running on a diverse range of systems from the ARM architecture to IBM Z/Architecture mainframe computers. The first port beyond Linux's original 386 architecture was to the Motorola 68000 platform by Amiga users, but to do this port they replaced major parts of the kernel, leading Torvalds to call it a fork and a "Linux-like operating system". With this experience in mind, Torvalds led a major restructure of the kernel code to facilitate the next port effort, to the DEC Alpha AXP platform, and support both 386 and Alpha in a single source tree.

Linux runs as the main operating system on IBM's Blue Gene supercomputers. As of June 2013, the Linux OS family has a 95.2% share of all systems on the TOP500 supercomputers list. Linux has also been ported to various handheld devices such as TuxPhone, Apple's iPod and iPhone. Some operating systems developed for mobile phones use modified versions of the Linux kernel, including Google Android, Firefox OS, HP webOS, and Nokia Maemo.

Kernel panic

In Linux, a "panic" is an unrecoverable system error detected by the kernel, as opposed to similar errors detected by user space code. It is possible for kernel code to indicate such a condition by calling the panic function located in the header file sys/system.h. However, most panics are the result of unhandled processor exceptions in kernel code, such as references to invalid memory addresses. These are typically indicative of a bug somewhere in the call chain leading to the panic. They can also indicate a failure of hardware, such as a failed RAM cell or errors in arithmetic functions in the processor caused by a processor bug, overheating/damaged processor, or a soft error.

Kernel oops

A report of a bug in the kernel is called an "OOPS". These crash reports are automatically collected and can be sent upstream by various software, such as kerneloops, ABRT (Fedora) and apport (Ubuntu). KernelOops.org collected these reports and published statistics on their website.

Security

Computer security is a much-publicized topic in relation to the Linux kernel, because a large number of bugs in the kernel can be potential security flaws, whether they allow privilege escalation or create denial-of-service attack vectors. Over the years, numerous such flaws were found and fixed in the Linux kernel. New security features are continuously implemented to address computer insecurity issues in the Linux kernel.

Critics have accused kernel developers of covering up security flaws or at least not announcing them. In response, in 2008, Linus Torvalds replied, "I personally consider security bugs to be just 'normal bugs'. I don't cover them up, but I also don't have any reason what-so-ever to think it's a good idea to track them and announce them as something special...one reason I refuse to bother with the whole security circus is that I think it glorifies—and thus encourages—the wrong behavior. It makes 'heroes' out of security people, as if the people who don't just fix normal bugs aren't as important. In fact, all the boring normal bugs are *way* more important, just because there's a lot more of them. I don't think some spectacular security hole should be glorified or cared about as being any more 'special' than a random spectacular crash due to bad locking."

At times, bugs have been corrected in Linux before other systems. In May 2012, a difference between the implementations of the SYSRET instruction in AMD and Intel processors was found to cause vulnerabilities in major systems such as Windows, FreeBSD, XenServer, and Solaris. The issue had been fixed in the Linux kernel since 2006.

Raw hardware devices are protected from direct access, and the file system has an inbuilt security system giving individual access to files on three levels, user only, group membership, and world access.

Long term releases with constant security updates are also available for most distributions. Reboot-less security updates can even be applied to the kernel by using technologies such as ksplice.

9.2 Feature history

Version 1.0 of the Linux kernel was released on 14 March 1994. This release of the Linux kernel only supported single-processor i386-based computer systems. Portability became a concern, and so version 1.2 (released 7 March 1995) gained support for computer systems using processors based on the Alpha, SPARC, and MIPS architectures.

Version 2.0 was released 9 June 1996. There were 41 releases in the series. The major feature of 2.0 was SMP support (that is, support for multiple processors in a single system) and support for more types of processors.

Version 2.2 (released 26 January 1999) removed the global spinlock and provided improved SMP support, and added support for the m68k and PowerPC architectures as well as new file systems (including read-only support for Microsoft's NTFS).

Version 2.4.0, released on 4 January 2001, contained support for ISA Plug and Play, USB, and PC Cards. It also included support for the PA-RISC processor from Hewlett-Packard. Development for 2.4.*x* changed a bit in that more features were made available throughout the duration of the series, including: support for Bluetooth, Logical Volume Manager (LVM) version 1, RAID support, InterMezzo and ext3 file systems.

Version 2.6.0 was released on 18 December 2003. The development for 2.6.*x* changed further towards including new features throughout the duration of the series. Among the changes that have been made in the 2.6 series are: integration of μClinux into the mainline kernel sources, PAE support, support for several new lines of CPUs, integration of ALSA into the mainline kernel sources, support for up to 2^{32} users (up from 2^{16}), support for up to 2^{29} process IDs (64-bit only, 32-bit arches still limited to 2^{15}), substantially increased the number of device types and the number of devices of each type, improved 64-bit support, support for file systems which support file sizes of up to 16 terabytes, in-kernel preemption, support for the Native POSIX Thread Library (NPTL), User-mode Linux integration into the mainline kernel sources, SELinux integration into the mainline kernel sources, InfiniBand support, and considerably more. Also notable are the addition of several file systems throughout the 2.6.*x* releases: FUSE, JFS, XFS, ext4 and more. Details on the history of the 2.6 kernel series can be found in the ChangeLog files on the 2.6 kernel series source code release area of kernel.org.

Version 3.0 was released on 22 July 2011. Torvalds announced that the big change was, "NOTHING. Absolutely nothing." 30 May 2011 saw Torvalds announce, "...let's make sure we really make the next release not just an all new shiny number, but a good kernel too." After the expected 6–7 week development process, it would be released near the 20th anniversary of Linux.

In December 2012, Torvalds decided to reduce kernel complexity by removing support for i386 processors, making the 3.7 kernel series the last one still supporting the original processor. The same series unified support for the ARM processor.

Version 3.11, released on 2 September 2013, adds many new features such as new O_TMPFILE flag for open(2) to reduce temporary file vulnerabilities, experimental AMD Radeon dynamic power management, low-latency network polling, and zswap (compressed swap cache).