Introducing the 3.3 and 3.4 Linux kernels

Review the latest Linux kernel and its features for Google Android, Open vSwitch, networking, and more

M. Tim Jones
Independent author
Consultant

Skill Level: Intermediate

Date: 19 Jun 2012

In March 2012, version 3.3 of the Linux kernel was released (followed in by version 3.4 in May). In addition to a plethora of small features and bug fixes, several important changes have arrived with these releases, including the merging of the Google Android project; merging of the Open vSwitch; several networking improvements (including the teaming network device); and a variety of file system, memory management, and virtualization updates. Explore many of the important changes in versions 3.3 and 3.4, and have a peek at what's ahead in 3.5.

Connect with Tim

Tim is one of our most popular and prolific authors. Browse all of Tim's articles on developerWorks. Check out Tim's profile and connect with him, other authors, and fellow developers in the developerWorks community.

The 3.3 and 3.4 releases of the Linux® kernel include an impressive feature set, but they're also something of an ominous milestone. The 3.3 release is the first Linux release to exceed 15 million lines of code (using an honestly flawed measure). If you subtract the variant portions of the Linux kernel (such as drivers, architecture-dependent code, and various tools), the number drops to under 4 million—a behemoth in its own right.

What's potentially ominous about this milestone is both the speed at which the Linux kernel is growing (50% growth since 2008) and whether this growth might begin to negatively affect the efficiency (both power and performance) of the Linux kernel. Power and performance don't tend to be measured on a per-patch basis, so a bug can easily creep into a released kernel and persist for some time (for example, the PCI Express [PCIe] Active State Power Management power issue that is fixed in 3.3 but existed in the kernel for a year).

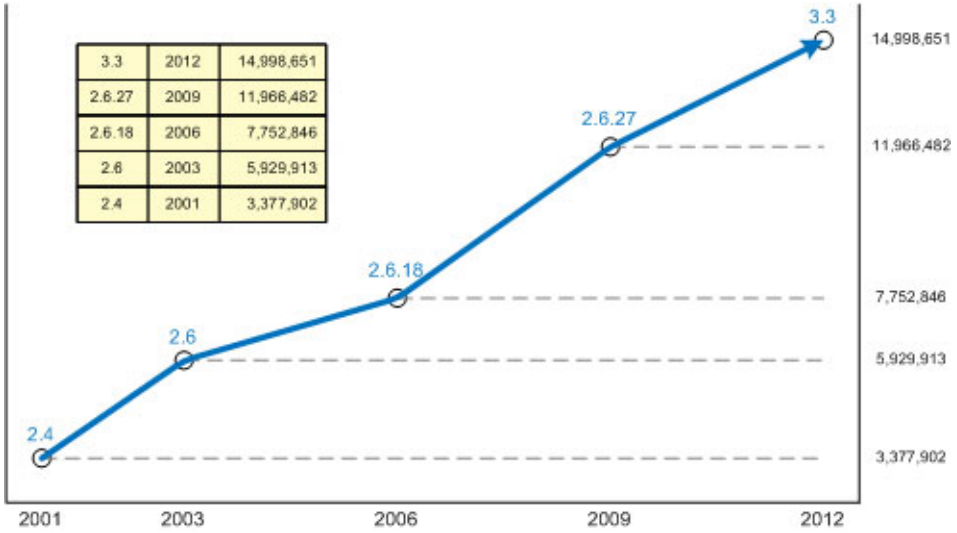
Linux kernel versioning

Kernels are versioned using a three-segment number scheme consisting of a kernel version followed by a major revision and minor revision (for example, 3.3.4). Release candidate kernels include an *rc#"* suffix that includes a number representing the particular candidate (for example, the current is 3.5rc1).

In less than 21 years, Linux has grown from just over 10,000 lines of code to more than 15 million. Although the majority of this code resides in the drivers subtree, the complexity of the kernel is increasing with its size. Someday soon, this expansion could result in changes to the kernel to remove the complexity and increase its maintainability.

As shown in Figure 1, the Linux kernel has grown quickly since the 2.4 release in 2001 (from 3,377,902 lines to 14,998,651 lines in 2012). Each year in that period, around a million lines of new code was added to the kernel. That's a staggering figure and should strike fear into any software developer.

Figure 1. Kernel sizes from release 2.2 (2001) to release 3.3 (2012)

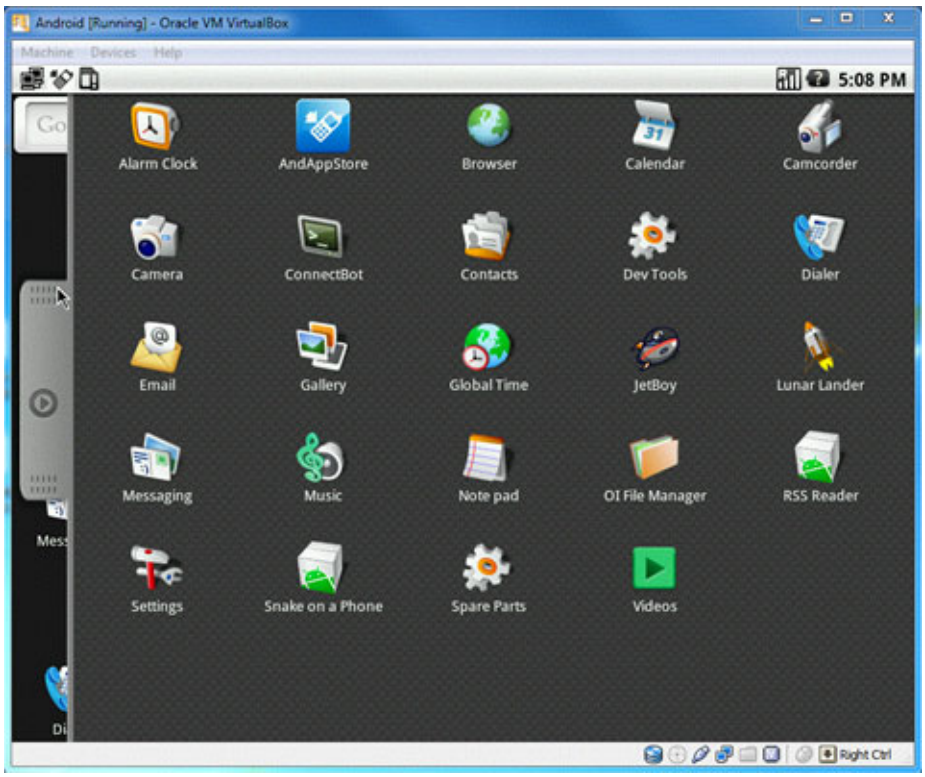


Torvalds himself is quoted as being concerned about future maintenance as the kernel grows. With around 4 million lines in the kernel proper, the current kernel management approach might need enhancement.

Android integration

The biggest news in the 3.3 kernel is the introduction of Google Android to the mainline kernel. This integration will continue in Linux 3.4, but enough of the Android fork is on the mainline to support booting the Android user space (see Figure 2). The Android kernel is a fork of the Linux kernel, with several additional features necessary for power and resource-efficient operation (as required by a power-constrained mobile device). Although the focus is on the ARM architecture, there's also support for x86 (such as is used in the Google TV project).

Figure 2. Booting to the Android user space with Android x86 on Oracle VM VirtualBox



Collaboration issues between Linux maintainers and Google led to Android being developed independently for a couple of years. The 2011-2012 winter saw the creation of the Android Mainlining Project, whose goal was to integrate Android drivers and features into the mainline Linux kernel. This work was introduced in the 3.3 release and will find further integration in the 3.5 release.

Android created several enhancements to Linux that were necessary to be competitive in the mobile environment. Examples include fast interprocess communication (IPC), improved application memory management, and a solution to the large contiguous physical memory management problem.

The driver called *Binder* is Android's answer to IPC. Android developers could have easily reused existing approaches, but Binder includes unique features that were not available (including zero-copy message passing and credential passing). Within Android, applications never exit, so they continue execution until the kernel removes them. The Shrinker exists as a mechanism to improve memory utilization as it gets low. Applications register a function that is called to minimize memory, and the kernel calls these functions when memory gets tight. One other addition by Android is Pmem, which provides the capability to allocate large physically contiguous buffers when needed (such as those required for a camera function). Pmem exports a user space driver for this type of memory allocation. Other capabilities have also been integrated but are specific to the mobile domain.

Certain features have not yet made their way into the kernel, such as *wakelocks*—a power-management feature that allows a component to prevent the system from entering a low-power state (for example, if an update is occurring). The absence of wakelocks won't prevent an Android system from booting, but it will drain the battery quickly.

As Android is merged back into the Linux kernel, it's another great illustration of the flexibility of the Linux kernel (from embedded systems and mobile devices to the largest mainframes and supercomputers). With more than 300 million Android devices in use right now, Linux continues its evolution as a universal platform.

Open vSwitch

Linux continues to be the platform of choice for virtualization. In addition to being a world-class operating system, Linux doubles as a world-class hypervisor. The mainlining of the Open vSwitch furthers this status by providing an out-of-the-box experience for virtualization and Infrastructure as a Service (IaaS) users.

A virtual switch is nothing more than a soft version of a physical switch. Recall that platform virtualization (as implemented by Kernel-based Virtual Machine [KVM] or Xen) allows you to run multiple operating system instances (as VMs) on a hypervisor that carves the physical platform into various virtual platforms. The introduction of a virtual switch extends this abstraction by introducing virtual forms of networking infrastructure. A virtual switch provides an efficient means for VMs to communicate with one another over a virtual network. Open vSwitch extends this abstraction across virtual hosts, allowing VMs on one physical host to transparently communicate with VMs on another physical host.

Within Open vSwitch, you'll find a rich set of features for virtual networking, including quality of service, virtual LANs, traffic filtering and isolation, and a variety of monitoring and control protocols (such as OpenFlow and NetFlow). Although Linux had an existing virtual switch implementation (called the *Linux Bridge*), the Open vSwitch is a far more feature-rich solution (including multihost management) and therefore is a welcome addition. See Resources for more information.

File system changes

Kernel release 3.3 has several file system changes across a number of file systems for both users and developers. For users, an online resize is available for the fourth extended file system (ext4) through an I/O control (where *online* means that the system remains operational). This means that the entire resize is performed in the kernel, which results in a much faster resize.

For the B-tree file system (Btrfs), the balance operation (which is used to change the underlying structure of metadata, such as if a new drive was added) has been rewritten, supporting pause and resume. Btrfs enhancements continued in release 3.4 with a new data-recovery tool (btrfs-restore), which can be used to extract files

from a damaged btrfs file system. In addition, in 3.4, Btrfs has several performance improvements and improved error handling (including removal of panics, replacing them with graceful error management). Prior to 3.4, Btrfs did not play well as a file system within a VM because of its copy-on-write mechanisms. Tuning has been performed to minimize these disruptions.

Software redundant array of independent disks (RAID) was also updated to support hot-replace, which allows data from a volume (marked replaceable with `mdadm`) to be migrated to another volume so that the original can be removed. Finally, 3.4 has added read-only support for the QNX4 file system.

For developers, it's now possible to inject errors into Network File System to test the client's ability to recover from them (through sysfs). For Btrfs developers, a new utility for integrity checking has been added that can be used to identify invalid Write requests and should help resolve bugs more quickly.

Networking enhancements

As Linux is on the leading edge of networking capabilities, several enhancements are available in kernel release 3.3.

For low-latency infrastructures (such as high-performance computing), a SCSI Remote Direct Memory Access (RDMA) protocol target driver has been integrated. Secure Remote Password is a protocol that allows you to use RDMA as the underlying transport for block storage devices. This particular addition allows Linux to expose a block device using SRP through which remote initiators can attach for block I/O. RDMA is supported by InfiniBand and is quite common in high-performance clusters.

The Random Early Detection (RED) packet scheduler was modified using a new algorithm from Sally Floyd, Ramakrishna Gummadi, and Scott Shenker called *Adaptive RED.* RED has proven to be an efficient packet scheduler algorithm (which drops packets it cannot buffer as a function of available queue size) but has been found to be sensitive to the level of congestion within a network. RED treats the queue size as a probability for packet drop, so in an empty or near-empty queue, all packets are accepted, but as the queue becomes full, all packets are dropped. Adaptive RED dynamically alters the probability for drop by measuring how aggressively the algorithm has behaved for dropping packets. You can read more about this algorithm in the Adaptive RED paper, for which Resource provides a link.

A new teaming network device has been added that replaces the older kernel bonding driver. The teaming device allows for the creation of virtual interfaces that aggregate the bandwidth available from multiple physical Ethernet devices (as defined by link aggregation and 802.1AX). This device can be used for network performance (aggregating multiple physical devices) or to provide redundancy (transparent failover). Two modes are currently supported, allowing traffic to simply

be allocated round robin across the physical ports, or a port can be defined as an active backup to route all traffic in the event the primary network connection is lost.

Among the large number of networking enhancements, one other interesting change is the addition of TCP buffer limits for control groups (or *cgroups*). Cgroups are used in a variety of ways, such as for isolating resources to a VM. This change allows both user space memory and kernel memory (such as is used for TCP buffers) to be tracked within a cgroup for better management of system resources.

Other interesting changes

Linux 3.3 also introduces changes that aren't specific to file systems or networking. In terms of new architectures, the Texas Instruments C6x processor is now supported directly (instead of as a separate project). The C6x is a single and multicore digital signal processor that is Very Long Instruction Word based but lacks modern features such as symmetric multiprocessing and cache coherency; it also lacks a memory management unit (MMU). Despite these architectural omissions, the C6x series are capable, with a rich set of peripherals and on-chip accelerators (security, fast Fourier transform, and so on). The 3.4 release supports the latest GPU processors such as Nvidia's Kepler and AMD's latest Radeon and Trinity releases.

The ARM architecture subtree now uses the large physical address extension as well as the introduction of support for the Nvidia Tegra 3 service on a chip, which is ideal as ARM challenges Intel in the low-power server space. Also available in 3.3 are several improvements to the AMD I/O MMU implementation, which improves the management of varying page sizes as well as increased security for devices (grouping or isolation of devices). Further, the Virtual Function I/O enhances KVM's ability to map devices to KVM guests. Finally, the S390 architecture was updated to support access to up to 64TB of RAM (beyond the prior limit of just under 4TB).

The Performance Monitoring Unit (PMU) is now virtualized for KVM, so guests can now access a PMU for their virtual platform. This exposes several useful performance events to each guest, including retired instructions, cache references and misses, and branch instructions executed and missed. Another useful virtualization feature for Xen is support for secure discard. *Secure discard* means that the sector in question will be permanently removed instead of simply marked as free. Finally, the various virtual I/O drivers (blk, net, balloon, and console) now support Advanced Configuration and Power Interface S4 sleep state, meaning that guest VMs can hibernate on Xen.

For memory corruption issues, which can be tedious to debug, a new configuration item called `CONFIG_DEBUG_PAGEALLOC` has been added. This change checks CPU accesses to unallocated pages and can result in some loss of performance.

Looking ahead

Linux continues to move ahead, and with 3.4 available, the 3.5 candidate releases soon come to a close around August 2012. Linux 3.5 includes several new and interesting features.

Btrfs continues to be enhanced, this time with optimized write-back handling within the file system. The standard Linux file system (ext4) has also been enhanced with the ability to add checksums to metadata to help identify data tampering. Linux may soon support SCSI targets over FireWire or the USB Attached SCSI Protocol. Finally, user-space probes will be supported (for use with SystemTap to analyze user-space program behavior). Many more changes are expected as the release candidate evolves toward an August release.

Resources

Learn

- Go to the Linux Kernel Archives for the source of the most recent (and not so recent) kernels.
- Linux Kernel Newbies is a great source of information on kernel releases, including summaries of some of the prominent features in a given kernel release. You can learn more in the Linux 3.3 summary page.
- Google created the Android Open Source Project to develop and maintain the Android platform. The project also maintains a compatibility program, which is free and ensures that third-party developers do not develop incompatible Android implementations. You can learn more about Android at the Wikipedia page.
- You can emulate Android for a mobile device using the Android Emulator as well as emulate Android for Google TV Emulator (using Linux and KVM). The Google TV Emulator supports both ARM emulation and less-efficient x86.
- Android introduced several new components that have made their way into the Linux kernel, including the IPC Binder (fast IPC), Low Memory Killer, and Pmem (similar to the excluded Ashmem component).
- Linux powering Android is a great example of Linux as a flexible and universal platform. Find more examples in Look at Linux, the operating system and universal platform (M. Tim Jones, developerWorks, March 2012).
- Open vSwitch is a multilayer virtual switch that is now mainlined into the Linux kernel. This means that Linux as a hypervisor can now use the Open vSwitch out of the box. This is a welcome addition for Linux virtualization power users. To learn more about Open vSwitch, read this short introduction on the product, Why Open vSwitch. You can also learn more about the prior solution, called the Linux Bridge from The Linux Foundation.
- Read Anatomy of an open source cloud (M. Tim Jones, developerWorks, March 2010) for an introduction to virtual switching in the context of cloud computing and IaaS.
- Virtual networking in Linux (M. Tim Jones, developerWorks, October 2010) provides an introduction to the various forms of virtual switching in Linux, including Open vSwitch. You can also learn more about the various types of virtualization (all supported by Linux) in Virtual Linux (M. Tim Jones, developerWorks, December 2006) and the transformation of Linux into a hypervisor in Anatomy of a Linux hypervisor (M. Tim Jones, developerWorks, May 2009).
- Btrfs is a file system developed by Oracle to challenge ZFS from Sun. Btrfs can be used to build massive enterprise storage systems, with features like online repair and simplified maintenance. You can learn more at the Btrfs wiki page.
- Virtio is an I/O virtualization framework for Linux hypervisors. Virtio provides an efficient abstraction for hypervisors using a common set of I/O virtualization

drivers. Learn more in Virtio: An I/O virtualization framework for Linux (M. Tim Jones, developerWorks, January 2010).
- Writing RDMA Applications on Linux from Roland Dreier at Cisco provides a great introduction to RDMA and developing applications that use RDMA within the Linux environment.
- Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management provides a detailed treatment for the algorithmic changes to the standard RED algorithm.
- Control groups are covered by Red Hat in its online RHEL documentation. You can also learn more about the application of TCP buffer limits to cgroups in the post, Per-cgroup TCP buffer limits at Linux Weekly News.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
- The Open Source developerWorks zone provides a wealth of information on open source tools and using open source technologies.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks. You can also follow this author on Twitter at M. Tim Jones.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.
- Get involved in the developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

M. Tim Jones

M. Tim Jones is an embedded firmware architect and the author of Artificial Intelligence: A Systems Approach, GNU/Linux Application Programming (now in its second edition), AI Application Programming (in its second edition), and BSD Sockets Programming from a Multilanguage Perspective. His engineering background ranges from the development of kernels for geosynchronous spacecraft to embedded systems architecture and networking protocols development. Tim is a platform architect with Intel and author in Longmont, Colo.