The Linux kernel

Past, Present and Future - the Linux way

Muli Ben-Yehuda

mulix@mulix.org

IBM Haifa Research Labs

The Linux Kernel

- linux is a free (speech and beer) UNIX like operating system, developed by thousands of volunteers around the world - including yours truly
- started as a hobby, grew and matured with the years
- countless man years went into it, some paid, some for fun
- one of the leading server operating systems today . . .
- . . . and one of the leading embedded operating systems
- poised to take over the desktop? Hebrew support?

The beginning

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Date: 25 Aug 91 20:57:08 GMT

Hello everybody out there using minix -
I’m doing a (free) operating system (just a hobby, won’t be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I’d like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things). I’ve currently ported bash(1.08) and gcc(1.40),
and things seem to work.This implies that I’ll get something practical
within a few months, andI’d like to know what features most people
would want. Any suggestions are welcome, but I won’t promise I’ll
implement them :-)
Linus (torvalds@kruuna.helsinki.fi)
PS. Yes - it’s free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that’s
all I have :-(
```

Linux - kernel or system?

Linux is the name given to the Linux kernel by its creator, Linus Torvalds. In common usage, Linux also refers to the entire suite of software installed on a machine, including libraries and end user applications, most of which came from the GNU project. How do we know which is which? by context.

As a side note, there are “Linux distributions” that do not use the Linux kernel, and at least theoretically, the Linux kernel could be used by other operating systems.

How does it work?

- UNIX like kernel, exposing POSIX interfaces to user space (system calls), but only where POSIX isn't braindead. We can aford to implement things our own way if they give substantial performance improvement
- complete from-scrach implementation of a monolithic kernel
- emphasis on pragmatic solutions - Linux is not a toy or research OS! Having said that, we play with it plenty ;-)
- most of the kernel is platform independent, supports many different platforms and hardware flavors
- same codebase runs on everything from wrist watches to super computers

How is it developed?

- (there used to be) a stable branch and experimental branch - 2.4 was the last stable branch. With 2.6, the kernel maintainers are experimenting with various new schemes that blur the distinction between stable and experimental).
- (there used to be) **long** formal release cycles, but closely adhering to the “release early, release often” principle for development snapshots
- many people, companies and organizations, each pushing to have their prefered patches merged. Only one person can actually merge things - Linus Torvalds, a benevolet dictator
- Linus’s job is to say **NO**
- IBM contributes heavily to Linux development, via the LTC and OSDL

The past

- Linux 1.0 - March 1994, i386 only, UP only
- Linux 1.2 - March 1995, no longer i386: Alpha, Sparc and Mips are now supported
- Linux 2.0 - June 1996, initial SMP support (guess who sponsored the work...)
- Linux 2.2 - January 1999, many improvments all over the system. Considered one of the better Linux kernel trees, many people are still using it.

The old stable branch

Linux 2.4 is the latest stable release tree. It is in wide usage and obviously works, but has its share of problems and flaws (obviously, those are fixed in the latest and greatest, 2.6)

- highly divergent vendor trees. The Linux vendors, such as redhat and SuSE, are under a lot of customer pressure and thus have a different view of what should go in the kernels they distribute.
- development of the 2.4 kernel tree has ground to an almost standstill. (Almost) all new development occurs in the 2.5 kernel.
- having said that, most(?) commercial kernel code is still being written on 2.4 and even 2.2

2.6

Linux 2.6

VM Changes

- the actual ’reverse mappings’ part of Rik van Riel’s rmap vm was merged. The VM paging algorithm should be smarter now, and VM behaviour under certain loads should improve. Slight performance hit due to copying the pte chains on fork.
- what is rmap? rmap allows the VM to know, given a physical memory page, which processes are using it - more intelligent decisions on heavy load and swapout.
- better behaviour under load, better behaviour on “enterprise machines”. The kernel booted with 32Gb and 64Gb of memory on 32 bit machines! (work done mostly by LTC members)

Kernel Preemption

- wsers should notice much lower latencies especially in demanding multimedia applications.
- code which is SMP safe should mostly be preempt safe. But, there are still cases where preemption must be temporarily disabled where we do not. These areas occur in places where per-CPU data is used.
- several subsystems are not known to be preempt safe - be careful when enabling preempt.
- if you get “xxx exited with preempt count=n” messages in syslog, don’t panic, these are non fatal, but are somewhat unclean. (Something is taking a lock, and exiting without unlocking)

Scheduler Improvements

- Ingo Molnar reworked the process scheduler to use an O(1) algorithm.
- users should notice no changes with low loads, and increased scalability with large numbers of processes, especially on large SMP systems.
- utilities for changing behaviour of the scheduler (binding processes to CPUs etc). http://tech9.net/rml/schedutils.
- sched_yield() and yield() can now make you sleep for a *long* time.
- 2.5 adds system calls for manipulating a task's processor affinity: sched_getaffinity() and sched_setaffinity()
- work on the scheduler, especially in the area of interactive performance, is on-going

Threading improvements

- lots of work went into threading improvements. Some of the features of this work are:
  - generic pid allocator (arbitrary number of PIDs with no slowdown, unified pidhash).
  - POSIX thread signals stuff (atomic signals, shared signals, etc.)
  - threaded coredumping support
  - sys_exit() speedups (O(1) exit)
  - generic, improved futexes
- users should notice is a significant speedup in basic thread operations this is true even for old-threading userspace libraries such as LinuxThreads.
- Native Posix Threading Library (NPTL). A userspace threading library utilizing the new threading improvements.

IO subsystem

- considerable throughput improvements over 2.4 due to much reworking of the block and the memory management layers.
- assorted changes throughout the block layer meant various block device drivers had a large scale cleanup whilst being updated to newer APIs.
- O_DIRECT improvements, size and alignment of data per device, not filesystem.
- block devices can now access up to 16TB on 32-bit architectures, and up to 8EB on 64bit architectures.
- AIO is now included in the kernel, but seems to offer no substantial benefits?

OProfile

A system wide performance profiler has been included in 2.5. With this option compiled in, you'll get an oprofilefs filesystem which you can mount, that the userspace utilities talk to. The userspace utilities for this are very young, and still being developed. You can find out more at
http://oprofile.sourceforge.net/oprofile-2.5.html
Oprofile is pretty useful, from experience. It allows you to profile both userspace applications and the kernel (hence the "system wide" moniker).

what is UML?

User Mode Linux (UML, hereafter) is a port of Linux (the kernel) to run as a program inside Linux (the system). Instead of working directly with the hardware, UML uses the host’s system call interface in place of the hardware. Surprisingly enough, it actually works.

what is it good for?

- testing and debugging kernel patches, without requiring a reboot
- private servers on shared hosts
- experimenting with system administration scenarios
- teaching operating systems ;-)
- UML clusters...
- (when UML-win32 comes of age) running Linux on windows machine - tapping into unused resources at night

how does it work?

- A UML process executes a system call instruction (int 0x80)
- via ptrace, the tracing thread is woken up
- the tracing thread annuls the system call on behalf of the UML process, and then forces the kernel to execute the system call
- the system call is executed, and when it is done, the tracing thread is woken up again
- the tracing thread manipulates the UML process state to think it completed the system call
- the UML process continues

more goodies

(In no particular order) Infiniband, SELinux, Ingo’s RT work, seccomp, 4K stacks, auditing framework, kernel events, laptop mode, IO schedulers, sched domains, circular pipe buffers, CKRM, kprobes, kexec, debugfs, four-level page tables, inotify, futexes, fuse, reiser4, NX support, address space randomization, netpoll, VIRTUALIZATION.

more info

The Linux kernel:

- http://www.kernel.org
- Understanding the Linux Kernel, by Bovet and Cesati, 2nd edition
- the linux-kernel mailing list - http://marc.theaimsgroup.com/?l=linux-kernel
- the kernelnewbies community - http://www.kernelnewbies.org

What’s new in 2.6:

- http://www.codemonkey.org.uk/post-halloween-2.5.txt

User Mode Linux:

- http://user-mode-linux.sf.net

Q, A, M and T?