The kernel report

(Korea Linux Form 2012 edition)

Jonathan Corbet
LWN.net
corbet@lwn.net


October, 2011

A slow moment at the Kernel Summit

The 3.1 kernel

October 24, 2011
(8,693 changesets,
1,168 developers)

A 95 day cycle

Since then...

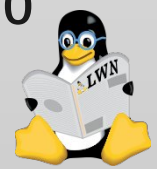
61,000 changesets merged
2998 developers have contributed
390 employers have contributed
The kernel is 1.24 million lines bigger

Recent release history

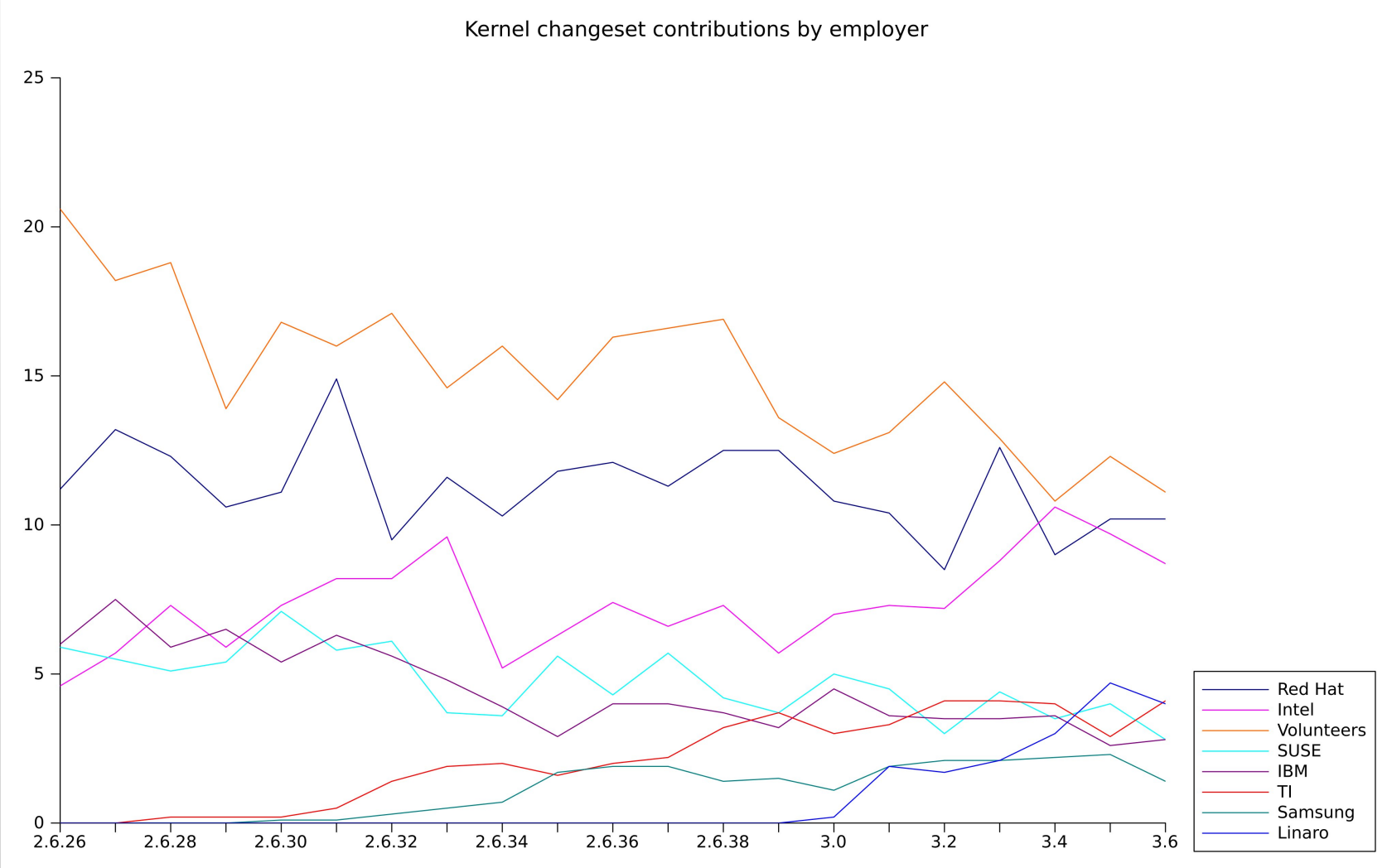
Release	Date	Days	Csets	Devs
3.1	Oct 24	95	8,693	1,168
3.2	Jan 4	72	11,828	1,309
3.3	Mar 18	74	10,550	1,247
3.4	May 20	63	10,899	1,286
3.5	July 21	62	10,957	1,195
3.6	Sep 30	71	10,247	1,216

Most active employers (3.1-3.6)

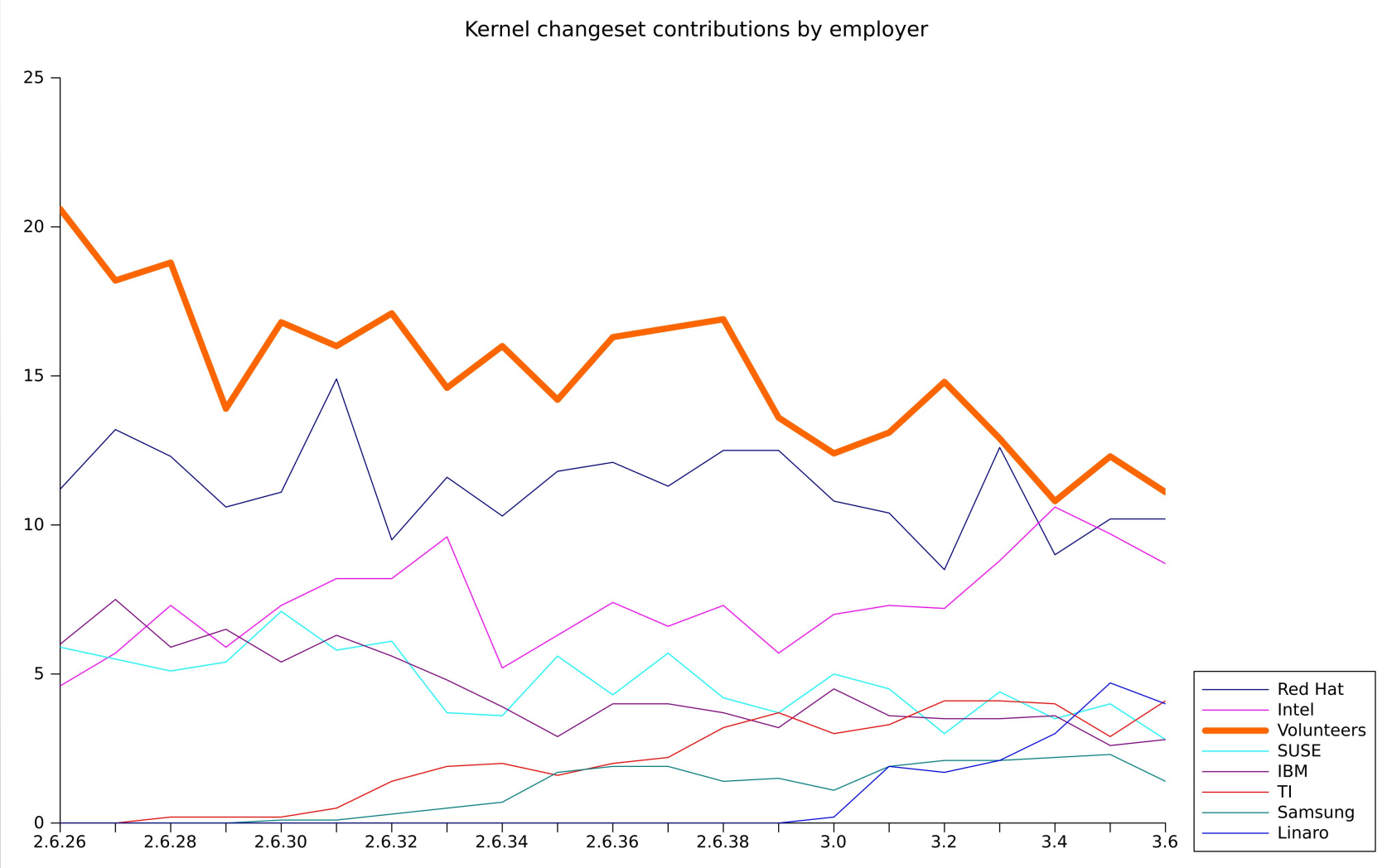
(None)	13.1%	Broadcom	1.9%
Red Hat	10.2%	Samsung	1.9%
Intel	8.9%	Ingics Tech	1.7%
(unknown)	5.7%	Qualcomm	1.7%
Texas Inst.	3.8%	Oracle	1.7%
SUSE	3.5%	Freescale	1.2%
Linaro	3.3%	Vision Engraving	1.2%
IBM	2.9%	NVidia	1.1%
Wolfson Micro	2.2%	Wind River	1.0%
Google	2.1%	Linux Foundation	1.0%
consultants	2.1%	AMD	1.0%



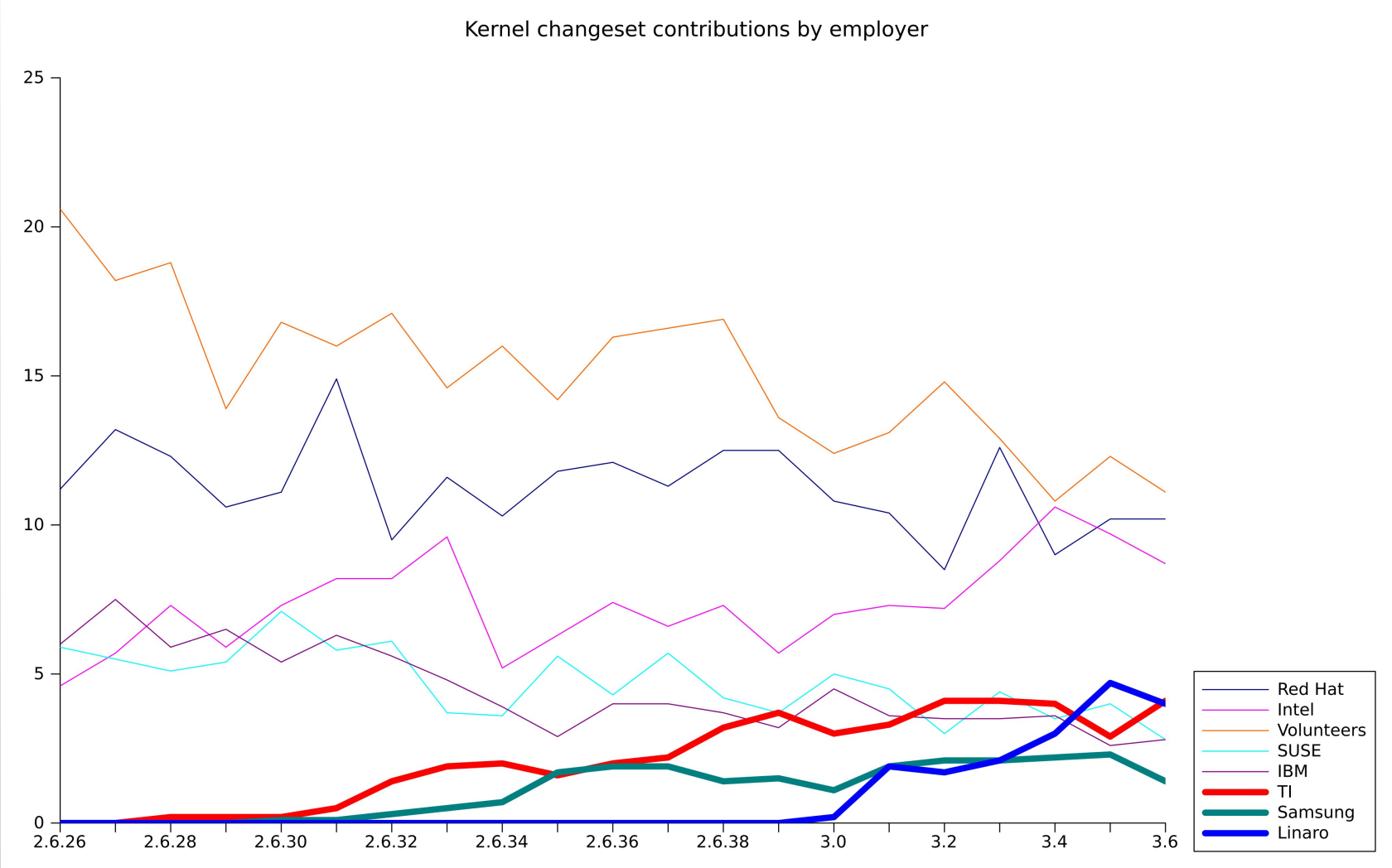
Employer participation since 2.6.26



Volunteer participation



Mobile/embedded participation



Stable updates

Most users do not run mainline kernels

- The run distributor kernels...
- ...that are built on the stable series

Stable updates

- Get fixes to users after a mainline release

Stable updates

Every kernel supported for one development cycle
    (3.5 reaching end of life now)

Occasional kernels for a two-year period
    Long-term support initiative
    3.0, 3.4 currently

All done by Greg Kroah-Hartman

Other long-term kernels

3.2.x

Ben Hutchings

For as long as Debian 7.0 is supported

2.6.32

Willy Tarreau

Infrequent updates

Lots going on

What has been accomplished...

...and what's coming?

The 3.7 kernel

Due in early December

Features

- 64-bit ARM support
- Xen on ARM
- Supervisor mode access prevention
- TCP fast open (server side)
- IMA integrity extension

Networking

Performance/protocol work

- TCP proportional rate reduction [v3.2]
- TCP fast open [v3.6, v3.7]
- TCP friends

Feature work

- Near-field communications (NFC) [v3.1]
- Network priority controller [v3.3]
- OpenVswitch [v3.3]
- TCP buffer size controller [v3.3]
- TCP connection repair [v3.5]
- IPv6 NAT [v3.7]

Bufferbloat work

Byte queue limits [v3.3]
- Keep device queues small

CoDel queue management [v3.5]
- Let router queues drain

TCP small queues [v3.6]
- Minimize in-flight data in the stack

Ongoing
- Debloat drivers and subsystems
- Continued queue management work
- Wireless issues

Embedded Linux

Lots going on

- ARM architecture
- Android integration
- ...

The ARM architecture

...poses some challenges

- Not really a platform
- Lots of competing vendors
- Short product cycles

Somebody needs to get a grip in the ARM community. I do want to do these merges, just to see how screwed up things are, but guys, this is just ridiculous. The pure amount of crazy churn is annoying in itself, but when I then get these "independent" pull requests from four different people, and they touch the same files, that indicates that something is wrong.
– Linus Torvalds, March 17, 2011

The ARM mess

Lots of code from lots of vendors

This is exactly what we had wished for!

- Just needed some more coordination

Fixing ARM

New ARM maintainer oversight
- Get ARM developers working together

Lots of code consolidation
- Fix per-platform duplicated code
- Good common abstractions
  - (Pin control, regmap, regulator, …)
- Move toward device tree
- Move drivers out of ARM architecture code

Interesting ARM work

big.LITTLE

Hybrid multiprocessing with diverse ARM cores

AARCH64

The 64-bit ARM processor [v3.7]

Single zImage

One kernel to bind them all and rule them

And, of course

...lots of new SoC models with each release

Android integration

A significant fork of the Linux kernel
- ...but still rather over-dramatized

Most Android code in the kernel now
- In the staging tree

Exceptions
- Wakelocks — we have other solutions
- ION allocator — nobody has done it yet
- Netfilter changes

System security

Hardening efforts are continuous
- Removing information disclosure issues
- Link restrictions [v3.6]

Seccomp filters [v3.5]

Supervisor mode access prevention [v3.7]

IMA integrity extension [v3.7]

Security module stacking

UEFI secure boot

The idea: restrict booting to known systems

Good:

- Block boot-time malware
- Ensure system is running the expected code

Bad:

- Tool for system lockdown
- Various practical hassles

Current thinking (x86)

Sign a minimal bootloader with a Microsoft key

Boot arbitrary signed kernel from there

No plans for Linux-specific signing keys

The cost (1)

Linux is dependent on Microsoft's good will

This solution is x86-only

- Windows 8 ARM systems must be locked down

Only distributor-provided kernels will boot

The cost (2)

No arbitrary code running in kernel mode

Thus, no:

- kexec()
- User-space drivers doing DMA
- No user-space access to I/O ports or memory
- Unsigned kernel modules
- ...

Most of this is likely for v3.7.

Filesystems

Slowing down a little
- Settling with ext4 and btrfs

Ext4
- Large block support [v3.2]
- Metadata checksumming [v3.5]
- Snapshots
- Inline data

Btrfs
- Send/receive [v3.6]
- RAID 5/6

F2FS

Flash-friendly filesystem
- Posted Oct. 5

Aimed at high performance on NAND flash
- Large segment sizes
- Modified log-structured filesystem design
- Two garbage-collection algorithms

Scheduling - NUMA

Sched/NUMA

- Assign a “home node” to each process
- Keep processes and allocations at home
- Migrate if local allocations become difficult

Scheduling - NUMA

Sched/NUMA

- Assign a “home node” to each process
- Keep processes and allocations at home
- Migrate if local allocations become difficult

AutoNUMA

- Assume that processes will wander
- Relocate memory to a process's current node
- Expensive usage-tracking mechanism

Other scheduling issues

Workload-specific performance regressons
- 3.6 PostgreSQL regression

Deadline scheduling
- Has a new maintainer
- Should still go in someday

Power-aware scheduling
- What is the right policy?

Per-entity load tracking

Scheduling - realtime

The -rt patch set still exists

Grand plans to mainline most of it

...but actual movement is slow

Testing

Kernel problems tend to be:

- Hardware dependent
- Workload dependent

As a result: they are hard to find automatically

- That's what you all are for!

Can we do better?

Trinity
- Smart fuzz testing

MMTests
- Find memory performance regressions early

Linsched
- Simulate the scheduler with lots of workloads

Wu Fengguang's build-and-boot tester
- 44 credited bug reports for v3.6

xfstests
- Increasingly capable filesystem test suite

Control groups

The kernel subsystem developers love to hate

What we're really talking about

Control groups

Organize processes into groups

A nested hierarchy

- Indeed: multiple hierarchies

What we're really talking about

Control groups

Organize processes into groups

A nested hierarchy
- Indeed: multiple hierarchies

Controllers

Apply a policy to processes in a control group
- Scheduling
- block I/O
- memory use
- network priority
- CPU affinities
- ...

Control group issues

- Multiple hierarchies
  - Which group is a given process in?

Controller issues

Inconsistent (or nonexistent) hierarchy support
- Warnings added in v3.7

No coordination between controllers
- Hard to share infrastructure
- Interesting interactions

To be done:
- Regularize hierarchical behavior
- High-level oversight of all controllers

Containers

Run an isolated Linux on the host kernel

Needs:

- Proper control groups and controllers
- Lots of internal namespace work
- Checkpoint/restore
- Lots of distribution-level work

Lots more I could talk about

Transparent huge pages
Nonvolatile RAM
Block I/O controllers
ptrace() and uprobes
udev
Power management
Sandboxing
Tracing
Checkpoint/restore
...

Some general concerns

Regression tracking

Regressions are the most important bugs
- They break systems that work now

We no longer have any formal regression tracking

Regression tracking

Regressions are the most important bugs
- They break systems that work now

We no longer have any formal regression tracking

How do we know the kernel is getting better over time?

Complexity

Arguably nobody understands some parts of the kernel

Complexity

Hardware complexity

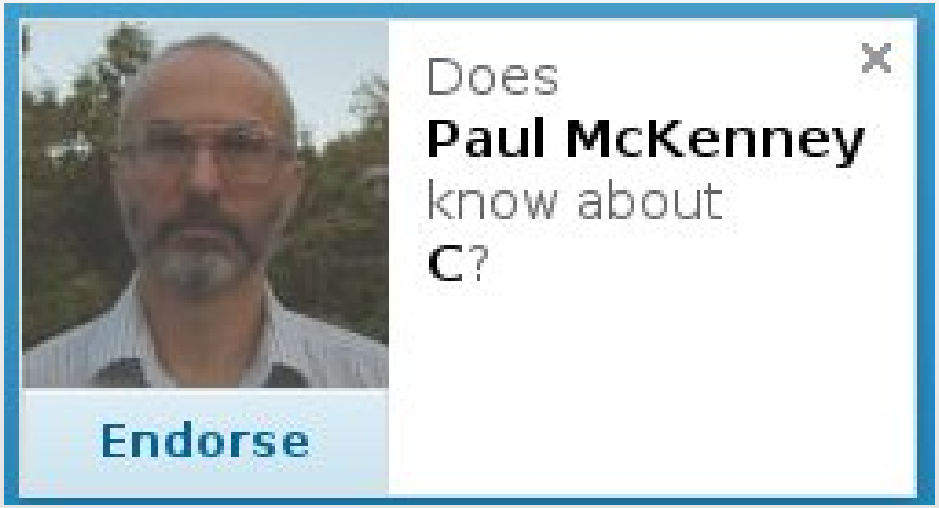
Software needs

- Scalability
- lockless algorithms

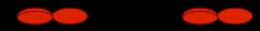
Complexity

Hardware complexity

Software needs
- Scalability
- lockless algorithms



Innovation



Interesting things are now done on Linux first

Proprietary forces

Software patents

- We all get radiated when it goes nuclear

Binary blobs

- Let the community help!

Locked-down hardware

- Let your customers play

I'm not worried about...

The health of Linux as a whole

Questions?