Linux Kernel Development:
Getting Started --
Activities & Recommendations

Randy Dunlap
rdunlap@xenotime.net

Linux Kernel Developer,
Maintainer, Mentor, and Janitor

LF-Japan Symposium
March, 2007

Agenda

- Timetable: began life as a 3-hour tutorial

- Just hitting highlights today, crash course

- Full presentation URL: various versions at:
  http://www.xenotime.net/linux/mentor/

- Abstract:

- Linux development is fast-paced and [as they say in Oregon] "things are different here." This tutorial introduces some of the Linux culture and how to succeed when working with the Linux development community.

Topics

- Open source development style, values, culture
- Linux rapid development cycle
- Testing

- Linux "maintainers" and hierarchy
- Communications methods

- Getting Involved
- Coding style
- How to submit Linux kernel patches
- Some best known practices

Major Goals

- Learn how to work successfully in the Linux kernel culture

- Learn ways to contribute to Linux kernel development

Development Style, Values, and Culture

- Learning curve, things are different

- Meritocracy – good ideas & good code are rewarded

- Chance to work on a real OS – any parts of it that interest you

- Massive amounts of open communication via email, IRC, etc. (i.e., not private)

Linux Culture

- Work in open, not behind closed doors (in smoke-filled rooms)

- Community allegiance is very high

- Do what is right for Linux

- Meritocracy: good ideas and good code are rewarded

- Often driven by ideals and pragmatism, bottom-up development

- Not driven by marketing requirements

- Don't just take, give back too:
  - Modifications are & remain GPL (if distributed)
  - Payment in kind, self-interest
  - Improve software quality, features used/understood more

Linux Culture (2)

- Committed to following and using standards (e.g., POSIX, IETF)

- Committed to compatibility with other system software

- Informal design/development: Not much external high-level project planning or design docs (maybe some internally at companies); can appear to be chaotic

- New ideas best presented as code, not specifications or requirements

- RERO: Release Early, Release Often -- for comments, help, testing, community acceptance

- Possible downsides: flames, embarrassment

Linux Culture (3)

- Development community is highly technical
- Motivated and committed, but since many are volunteers, treat them with respect and ask/influence them, don't tell
- Continuous code review (including security)
- Continuous improvement
- Have fun!! :)

- **Follow the culture**

Linux Development Values

- Scratch your own itch
- Weekenders -> big business
- Code, not talk
- Pragmatism, not theory
- Thick skin
- Code producer makes [most] decisions
- Pride, principles, ethics, honesty
- Performance

- Hardware & software vendor neutral
- Technical merit, not politics, who, or money
- Maintainability & aesthetics: clean implementation, not ugly hacks (coding style)
- Peer review of patches (technical & style)
- Contributions earn respect

Some Things to Avoid

- Patents, binary modules, NDA

- Proprietary benchmarks

- Huge patch files

- Adding more IOCTLs

- Marketing

- Design documents

- Mention of accomplishments outside of the open source world

- No patch rationale

- How do I intercept a system call (or replace a syscall table entry)?

- Making demands instead of requests

- This {driver / feature} must be merged, it's important to our company.

- Date or release version requirements

Some Good Terms to Use

- Simpler
- Deletes N lines of code
- Faster (with data)
- Smaller (with data)
- Here's the code....
- Series of small patches....

- Tested... (how many configs)
- Builds on 8 architectures

Development Cycle

- Moved from split "stable" (2.even) and "development" (2.odd) trees – caused delay and backport mania

- Now accepting development patches into the -mm patchset and moving them to the mainline kernel tree after a shakeout period (e.g., 2.6.11-mm3)

- 2.6.x kernel version cycle: make patches against Linus's tree (unless they only apply to some other tree or patchset)

- Time between 2.6.x releases, intermediate 2.6.x-rcN

- Nightly snapshots; automated builds of releases; commits mailing list

- 2.6.x.y stable kernel patches

Development Cycles

- Rapid development cycle, no timelines/schedules
- Only online documentation has a chance of being up-to-date
- Accommodate large changes and high rate of change without regressions
- Open discussion (mailing lists, archives, not private)
- RERO, facilitates testing on a large variety of platforms
- Maintainers available and accessible, don't disappear for long periods of time
- Test suites
- Bug tracking

Linux 2.6 Kernel Tree & Branches

mainline

| 2.6.19 | 2.6.20-rc1 | -rc2 | -rc3 | -rc4 | -rc5 | -rc6 | -rc7 | 2.6.20 | 21-rc1 |

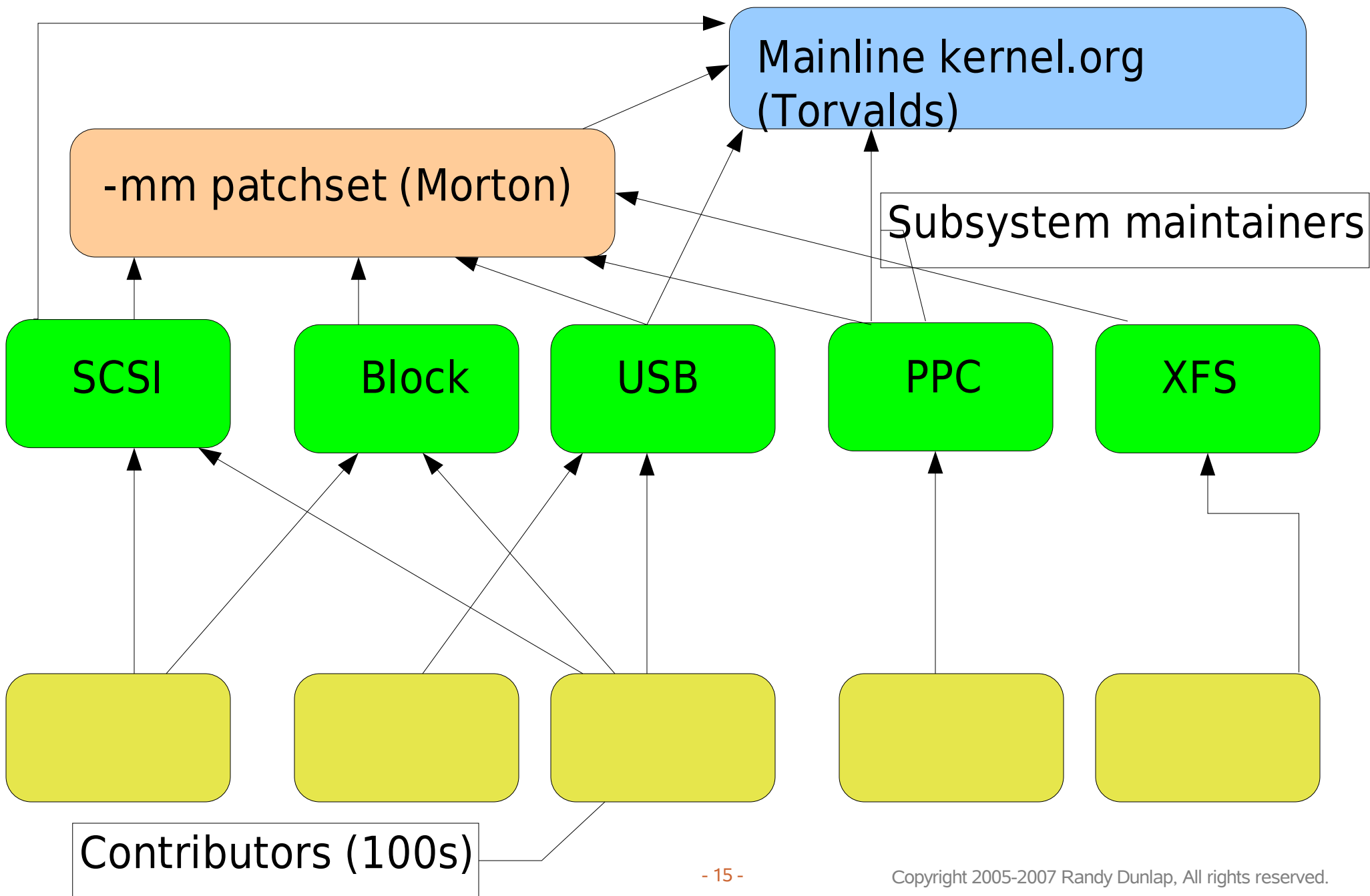**-mm patchset: review/test here before merge into mainline**

-mm1 -rc1-mm1 -mm1 -mm1 -mm1 X -mm1   -mm1

                                                     -mm2 -mm2

                                                     -mm3

stable patch series

| 2.6.19.1 | | .19.2 | | .19.3 .19.4 .19.5 |
| | | | | .20.1 |

Merges to Mainline (with exceptions)

Kernel Testing

- Part of the RERO culture

- You can contribute to Linux kernel development just by testing new kernels (stable and development branches or other trees that you are interested in, such as USB, ATA, ACPI, filesystems)

- Report bugs to the linux-kernel mailing list or project mailing lists or on http://bugzilla.kernel.org or (same) http://bugme.osdl.org

Kernel Testing Software

- LTP: http://ltp.sourceforge.net

- Open POSIX test suite:
  http://posixtest.sourceforge.net

- http://test.kernel.org frequent tests & reports

- Virtualization for testing: UML, qemu, bochs, XEN:
  see the Linux virtualization summary:
  http://www.linuxsymposium.org/proceedings/reprints/
  Reprint-Wright-OLS2004.pdf

Topics

- Open source development style, values, culture

- Linux rapid development cycle

- Testing

- Linux "maintainers" and hierarchy

- Communications methods

- Getting Involved

- Coding style

- How to submit Linux kernel patches

- Some best known practices

Maintainers and Hierarchy

- Loose hierarchy with "benevolent dictator"
- Kernel series maintainers (2.6) – Linus and Andrew Morton
- Patch ("stable") maintainers (2.6.x.y) – Greg Kroah-Hartman and Chris Wright
- Top-level maintainers are gatekeepers, integrators, tiebreakers or overrulers when needed
- Delegate to lieutenants and individual maintainers; share the load
- Strong trust system -> begin with small patches for credibility
- Lower-level maintainers don't have absolute authority

Maintainers & Hierarchy (2)

- Kernel Janitors, security kernels, some embedded support

- Arch and subsystem maintainers: coordinate subsystems and maintain consistency

- Driver maintainers: cover all current mainline kernels and update to new kernel APIs, even development APIs

- See files: linux/MAINTAINERS and linux/CREDITS

Communications

- Communicating is hard, let's go shopping
- Writing ideas/thoughts down is good (but too wordy may be ignored)
- Participate constructively
- Mailing lists & archives (newsgroups)
- Working in open/public (technical readers/writers) vs. embarrassment
- Discussion and decisions on lists, no meetings required
- Work through concensus (with exceptions)
- Project web pages, IRC channels (in References)
- Developer conferences

Mailing List Etiquette

- Use Reply-to-All, threaded (Message-ID, References)
  - > > Try A or B.
  - > I prefer A, sound OK?
  - yes
- Be prompt with replies (being responsive is important)
- No encoded or zipped attachments (inline preferred, text/plain attachments OK); others are often ignored
- No HTML or commercial email, no auto-replies (OOO/vacation)
- ALL CAPS == SHOUTING; rude, don't do it
- Use < 80-column width lines (70-72 is good) for text (not for patches)

Mailing List Etiquette (2)

- Keep it technical and professional. If attacked (flamed), stick with technical points, don't get involved with attacks, & move on.

- Trim replies (body) to relevant bits (don't modify To:/Cc: recipient list).

- Don't cross-post to closed mailing lists.

- Non-English speakers

- http://www.arm.linux.org.uk/armlinux/mletiquette.php

- RFC 1855: Netiquette Guidelines: http://www.ietf.org/rfc/rfc1855.txt

Mailing List Etiquette: No top-posting

- A: http://en.wikipedia.org/wiki/Top_post
  Q: Where do I find info about this thing called top-posting?
  A: Because it messes up the order in which people normally read text.
  Q: Why is top-posting such a bad thing?
  A: Top-posting.
  Q: What is the most annoying thing in e-mail?

  A: No.
  Q: Should I include quotations after my reply?

Development Conferences

- Linux Symposium (Ottawa, July)
- Linux Conference AU (LCA, usually March-April)
- LinuxTag (Germany, June)
- Linux Kongress (Germany, September)
- Kernel (September), GCC (June), Desktop (July) summits
- Focused mini-summits (networking, power management, storage management, filesystems, wireless, desktop)

Topics

- Open source development style, values, culture
- Linux rapid development cycle
- Testing

- Linux "maintainers" and hierarchy
- Communications methods

- Getting Involved
- Coding style
- How to submit Linux kernel patches
- Some best known practices

Getting Involved in Kernel Development

- Testing, feedback/results
- Learn some basics at http://kernelnewbies.org
- Find some small tasks that are identified at http://www.kerneljanitors.org
- Focus on an area that you are interested in (many to choose from)
- Can just fix compile/build/sparse problems as an introduction
- Fix bugs in the bugzilla database at http://bugzilla.kernel.org
- Add to kernel documentation

what is at http://kernelnewbies.org

- wiki, mailing list, IRC, FAQ
- Kernel hacking introductions, HOWTOs
- Tools
- Testing
- Tutorials
- Bugzilla
- Debugging tips and techniques
- Kernel Porting; Migrating 2.4 kernel code to 2.6
- Submitting kernel patches; upstream merge guide
- Kernel documentation

more at KernelNewbies.org

- Community Virtualization wiki:
  http://virt.kernelnewbies.org

- <u>Kernel Mentors</u> information:
  http://kernelnewbies.org/KernelMentors

- Kernel Mentors mailing list:
  http://selenic.com/mailman/listinfo/kernel-mentors

- "informal project to get experienced developers to mentor new developers and coach them on the best ways to get their code ready for submission"

- "The basic idea is matching new kernel developers up with experienced developers so they can get acquainted with existing best practices."

at http://www.kerneljanitors.org

- mailing list, IRC, FAQ, wiki, TODO list

- Documentation, like "Drivers DOs and DON'Ts", "How NOT to write a device driver", and "Writing Portable Device Drivers"

- Tools, Tips, and Tricks

- TODO list: always check to see if an item is still current before spending time on it

- The "other" KJ wiki: http://www.fsdev.net/wiki/
  - community wiki, not project maintainers
  - more recent, updated more often

Coding Style

- **Clean code, not for other OS-es or for other Linux versions**

- **Keep it looking like Linux code for readability, maintainability, debugging, etc.**

- Documentation/: CodingStyle, SubmittingPatches, SubmittingDrivers, SubmitChecklist, & web pages

How to Submit Linux Kernel Patches

- Patch -current mainline from kernel.org or -mm patchset
- Send patches to subsystem maintainer, driver maintainer, & mailing list
- Each patch (re-)submission should include feature justification and explanation, not just the patch
- Use the DCO ("Signed-off-by: Your Name your.name@example.com")
- Patches should be encapsulated (self-contained) as much as possible, not touching other code (when that makes sense)

Submitting Patches (2)

- ONE patch per email, logical progression of patches, not mega-patches, not attached and not zipped (cannot review/reply)

- Don't do multiple things in one patch (like fix a bug and do some cleanup)

- Check your email client: send a patch to yourself and see that it still applies (doesn't damage whitespace, line breaks, content changed) before going public with it

- Patch must apply with 'patch -p1'; i.e., use expected directory levels

- Don't use PGP or GPG with patches, they mess up patch scripts

Some Best-Known Practices

- Send patches directly to their intended maintainer for merging (they don't troll mailing lists looking for patches to merge)

- Copy patches to the appropriate mailing list(s), not private (don't work in isolation)

- Subscribe to relevant mailing lists (or use one representative for this)

- Listen to review feedback and promptly respond to it

Best Known Practices (2)

- Some maintainers do not acknowledge when they merge a patch; you just have to keep watching

- Use correct 'diff' directory level (linux/ top-level directory) and options (-up)

- Use source code to convey ideas

- Generate patch files against the latest development tree branch (-rcN) or mainline kernel if there is no current development branch

- Make focused patches or a series of patches, not large patches that cover many areas or that just synchronize a (CVS) repository with the kernel source tree

Best Known Practices (3)

- Use an email client that supports inserting patches inline (not as attachments)

- Begin with small patches: use kernel-janitor m-list

- For larger patches or complete drivers or features, use the kernel-mentors m-list (for beginner feedback/comments/corrections)

- Don't post private email replies to a public m-list (without permission)

- Don't introduce gratuitous whitespace changes in patches

Best Known Practices (4)

- Back up your patch with performance data (if applicable)

- Don't add binary IOCTLs unless there are no other acceptable options; use sysfs (/sys) or private-fs or debug-fs or relayfs or netlink if possible

- Make Linux drivers that are native Linux drivers, not a shim from another OS

- Don't introduce kernel drivers if the same functionality can be done reasonably in userspace

- Try to be processor- and distro-agnostic (except for CPU-specific code)

- Don't be afraid to accept patches from others

Best Known Practices (5)

- Keep your patch(es) updated for the current kernel version

- Resubmit patches if they are not receiving comments

- Release early, release often

- Open, public discussion on mailing lists

- One patch per email

- Large patches should be split into logical pieces and mailed as a patch series

- Make testing tools available & easy to use; your device(s) will get better testing

References & Backup

- Mailing lists
- Web pages

Related Documentation

- lwn.net articles: http://lwn.net/Articles/driver-porting/

- LDD3 book: http://lwn.net/Kernel/LDD3/

- Linux Kernel in a Nutshell: http://www.kroah.com/lkn/

- Greg (PCI, USB, driver core, sysfs maintainer): Coding Style, Writing Portable Code, et al (http://www.kroah.com/linux/)

References

- http://www.linuxsymposium.org/2005/ - OLS proceedings

- This full presentation: http://www.xenotime.net/linux/mentor/linux-mentoring-20

- Andrew (top kernel maintainer): TPP: The Perfect Patch: http://www.zip.com.au/~akpm/linux/patches/stuff/tpp.txt

- Jeff (net drivers maintainer): http://linux.yyz.us/patch-format.html

Mailing Lists for Linux Starters

- Trivial patch monkey: trivial@kernel.org and
  http://www.kernel.org/pub/linux/kernel/people/bunk/trivial

- http://vger.kernel.org/majordomo-info.html - has list info
  and taboos

- Kernel announcements:
  linux-kernel-announce@vger.kernel.org

Mailing Lists

- Most lists have spam filters [to get past]; you probably need to use them also

- LKML a.k.a linux-kernel (@vger.kernel.org)

- LKML FAQ at http://www.tux.org/lkml/

- Index: http://vger.kernel.org/vger-lists.html and their archives

- Kernel patch commits mailing list (for Linus/mainline): git-commits-head@vger.kernel.org

- MM-commits (Andrew Morton's patch tree): mm-commits@vger.kernel.org

More Kernel Project Mailing Lists

- Networking development: netdev@vger.kernel.org
- Index: http://oss.sgi.com/ecartis/
- Subsystems: arches, filesystems, MM/VM (http://www.linux-mm.org), security, drivers (ACPI [SF.net], I2C, IDE, video, PCI, PCMCIA, IEEE 1394 [SF.net], USB [SF.net], SCSI, Infiniband, Bluetooth)
- More mailing lists in MAINTAINERS file and at http://kernelnewbies.org
- TLF: http://lists.linux-foundation.org

Mailing List Archives

- Archives for almost all
  - http://gmane.org has interface
  - http://marc.theaimsgroup.com/ has many, with Search
  - http://lkml.org -- kernel list only
  - Google groups
- http://lwn.net/ -- summaries

Project Web Pages

- SourceForge.net (http://sf.net): web pages, mailing lists, CVS, bug tracking, etc.

- Linux Foundation: http://www.linux-foundation.org and http://developer.osdl.org

- Hardware vendors: IBM, HP, Dell

- Distro vendors (Red Hat, SUSE, Debian)

Linux kernel source tree References

- linux/ MAINTAINERS, CREDITS
- linux/Documentation/
  CodingStyle
  SubmittingPatches
  SubmittingDrivers
  feature-removal-schedule.txt
  stable_api_nonsense.txt
  SubmitChecklist
  HOWTO (do kernel development)

Credits

- Hugh Blemings
- James Bottomley
- Matt Domsch
- Jeff Garzik
- Clyde Griffin
- Christoph Hellwig
- Gerritt Huizenga
- Greg Kroah-Hartman

- Pat Mochel
- Andrew Morton
- Arjan van de Ven
- Ric Wheeler
- Cliff White
- Chris Wright
- Top-posting A&Q from a .sig on the old crackmonkey m-l