

# Fehlerfinden und Qualitätssicherung im Linux-Kernel

Paul Menzel (Max-Planck-Institut für molekulare Genetik)

10. März 2018

# Wer bin ich?



- ▶ Systemarchitekt beim Max-Planck-Institut für molekulare Genetik
- ▶ Diplom-Wirtschaftsmathematiker (TU Berlin)
- ▶ FLOSS-Befürworter

# Präsentation

Folien in Markdown mit Pandoc nach LaTeX-Beamer umgewandelt,  
verfügbar auf GitHub.

[https://github.com/paulmenzel/fehlerfinden\\_und\\_qualitaetssicherung\\_im\\_linux\\_kernel](https://github.com/paulmenzel/fehlerfinden_und_qualitaetssicherung_im_linux_kernel)

# Problem

- ▶ Hersteller/OEMs entwickeln nur für Microsoft Windows.
- ▶ Wenige Ausnahmen: Google Chromebooks und Apple, Purism(?), Dell
- ▶ Anderen Anbieter passen nur an
- ▶ Keine Ergebnisse für `git log --author=system76` und `git log --author=tuxedo`
- ▶ Mehr Motivation: Vortrag *Hilf mit, damit Linux nur besser und nie schlechter wird* von Thorsten Leemhuis, Sonntag 10 Uhr
- ▶ Hohe Änderungsraten und Testlabore erst im Kommen
- ▶ <https://intel-gfx-ci.01.org/>

# Arbeit mit Entwicklern

- ▶ Linux-Kernel in Distributionen meist nicht aktuell
- ▶ Linux-Kernel-Entwickler bevorzugen Meldungen bezüglich der aktuellen Version
  - ▶ Zweig *master* in Linus' Git-Depot
  - ▶ Entsprechender Entwicklungszweig

Neuer Linux-Kernel

# Pakete

- ▶ Ubuntu: <http://kernel.ubuntu.com/~kernel-ppa/mainline/>
- ▶ Fedora:  
[https://fedoraproject.org/wiki/Kernel\\_Vanilla\\_Repositories](https://fedoraproject.org/wiki/Kernel_Vanilla_Repositories)
- ▶ Debian: manchmal in *experimental*:  
<https://packages.debian.org/linux-image-4.16>
- ▶ andere Distributionen ähnlich

## Selber bauen (Debian)

```
$ sudo apt install kernel-package # Abhängigkeiten
$ git clone \
https://git.kernel.org/pub/scm/linux/kernel/git/
torvalds/linux.git
$ cd linux
$ cp -a /boot/config-4.15.0-1-amd64 .config
$ make help
$ make olddefconfig
$ make localmodconfig
$ make bindeb-pkg -j`nproc`
$ sudo dpkg -i ../linux*.deb
```



# Problemstellen

Bautests, Funktionen, ACPI, Energiemanagement (Power Management), Grafik, Audio, Netzwerk, Dateisystem, Geschwindigkeit, ...

# Bautests

- ▶ Mehr oder weniger gelöst: nur sehr selten Fehler beim Bauen.  
Danke Intel!
- ▶ Derzeit Tests mit Clang erwünscht und noch nicht gut abgedeckt

# Geschwindigkeit

- ▶ „Pet peeve“ (vergleiche mit Google Chromebooks, Apple-, MS-Windows-Geräte)

## Ziel

- ▶ Schneller Start des Linux-Kernels
- ▶ Schnelles Einschlafen und Aufwachen (ACPI S3)

[sleepgraph.py](#)/[bootgraph.py](#)

- ▶ <https://01.org/suspendresume>
- ▶ <https://github.com/01org/pm-graph>

*System analysis tool source code and binary, and a blog which gives updates on actual changes being made to the kernel using the tool*

# Praxis

- ▶ Füge `initcall_debug` zu Linux-Kommandozeile hinzu

```
$ sudo vim /etc/default/grub
```

```
$ sudo update-grub
```

```
$ systemctl restart
```

- ▶ Installiere `systemd-bootchart` mit `sudo apt install systemd-bootchart`
- ▶ `man systemd-bootchart`
- ▶ Füge `init=/lib/systemd/systemd-bootchart` zu Linux-Kernel-Kommandozeile hinzu

## systemd-bootchart

- ▶ SVG-Datei unter `/run/log/`
- ▶ Übersicht, welche Linux-Kernel-Funktionen, wie viel Zeit

## sleepgraph.py/bootgraph.py

- ▶ Kopie von letzter Veröffentlichung in Linux-Kernel-Quellen
- ▶ Nutze Git-Depot direkt:

```
$ git clone https://github.com/01org/pm-graph
$ cd pm-graph
$ sudo ./sleepgraph.py -c config/suspend-callgraph.cfg
```

- ▶ Standardmäßig eigenständiges Aufwachen nach 15 Sekunden
- ▶ Tiefe über Parameter `maxdepth` erhöhen

Beispiel mit Dell XPS 13 9370

# kseltest

- ▶ <https://www.kernel.org/doc/Documentation/kseltest.txt>
- ▶ <https://kseltest.wiki.kernel.org/>

*The kernel contains a set of “self tests” under the tools/testing/selftests/ directory. These are intended to be small tests to exercise individual code paths in the kernel. Tests are intended to be run after building, installing and booting a kernel.*

```
$ sudo apt install libcap-ng-dev libnuma-dev libfuse-dev  
$ make kseltest
```



Einschub: Fehlerberichte

## Einschub: Fehlerberichte

- ▶ LKML: Linux Kernel Mailing List
- ▶ <https://bugzilla.kernel.org/>
- ▶ Datei MAINTAINERS
- ▶ <https://01.org/linuxgraphics/documentation/how-report-bugs>

Grafik

# Grafik

- ▶ Komplex, viele Ebenen
- ▶ Framebuffer, DRM, Mesa, GPU-Dekodieren/-Kodieren
- ▶ Chamelium Board
- ▶ für Audio: Chamelium Audio Board

Grafik

## Grafik: igt-gpu-tools

- ▶ <https://cgit.freedesktop.org/drm/igt-gpu-tools/>
- ▶ früher intel-gpu-tools

```
$ git clone \
https://anongit.freedesktop.org/git/drm/igt-gpu-tools.git
$ cd igt-gpu-tools
$ # Installiere Abhängigkeiten (Debian)
$ sudo apt build-dep intel-gpu-tools
$ mkdir build
$ meson build
$ cd build
$ ninja
$ ninja test
```

## Grafik: weitere Testsammlungen

- ▶ <https://piglit.freedesktop.org/>

```
$ sudo apt install piglit
$ piglit run quick results/quick
[39129/39129] skip: 3169, pass: 35860, warn: 4, fail: 95, c
Thank you for running Piglit!
Results have been written to /home/joey/results/quick
```

- ▶ GPU-Testsammlung dEQP (drawElements Quality Program):  
<https://github.com/KhronosGroup/VK-GL-CTS>
- ▶ Proprietär: GFXBench

## Sanitizers



# UndefinedBehavior/AddressSanitizer

## CONFIG\_UBSAN

- ▶ UBSAN: Undefined behaviour in  
drivers/net/wireless/ath/ath10k/mac.c:3092:53: signed integer  
overflow

## CONFIG\_KASAN

- ▶ Re: BUG: KASAN: use-after-free in  
xhci\_trb\_virt\_to\_dma.part.24+0x1c/0x80

# Dateisysteme

# XFS

- ▶ `git clone`  
`git://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git`
- ▶ <https://stackoverflow.com/questions/21565865/filesystem-test-suites>

# Energieverbrauch

- ▶ PowerTOP

# SATA LPM/PSR

- ▶ SATA Link Power Management: Improving Linux laptop battery life: Testers Wanted
- ▶ Panel Self Refresh: Improving Linux battery life, enabling PSR by default, testers wanted

# Bluetooth

- ▶ kein externes Testgerät bekannt
- ▶ möglichst mit vielen Geräten testen (Hotel, . . . ); Sicherheit?

# Leistungsmessung (Benchmarking)

- ▶ Überschneidung mit Anwendungen (userspace)
- ▶ Beispiel: Phoronix Test Suite
- ▶ <https://openbenchmarking.org/>

Ausblick



- ▶ Hinzufügen von mehr Tests aus mehr Subsystemen

# Datenbank und Hochladen

- ▶ Übergreifende Datenbank mit Ergebnissen nicht bekannt
- ▶ Modell? → Seite mit Verweisen zu einzelnen Subsystemen (Meta-Suchmaschine)
- ▶ Riesige Datenmengen

# Fazit

1. Jeder kann mitmachen.
2. Distributionen sollten Programme zur einfachen Installation paketieren und ausliefern.
3. Paketarchive mit „Linux-Debug-Kernel“ zur einfachen Installation.
4. Live-Abbilder/Images zur einfachen Nutzung
5. Hersteller in die Pflicht nehmen, ihre Produkte ordentlich zu testen.
6. Entwicklung von Testgeräten und Entwicklern zur Verfügung stellen.

# Noch mehr Tests

1. Nicht nur Linux-Kernel auch andere Ebenen und Anwendungen mit Tests
2. aktueller Artikel zu Cairo

Fragen