

Praktikum angewandte Systemsoftwaretechnik (PASST)

bisect / upstream / Aufgabe 4

17. Mai 2018

Stefan Reif, Peter Wägemann, Florian Schmaus, Michael Eischer,
Andreas Ziegler, Bernhard Heinloth und Benedikt Herzog

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Fehlersuche mittels Bisektion

- Werkzeug zur Fehlersuche in einem git-Repository
- „Welche Revision hat den Fehler eingeführt?“
- Voraussetzungen:
 - Eine Revision in der der Fehler auftritt (üblicherweise die aktuelle)
 - Eine Revision in der der Fehler (noch nicht) auftritt
 - Manueller oder automatischer Test für den Fehler
 - Möglichst viele Revisionen sollten testbar sein (z.B. compilieren)

Bisektion (1/2)

- Mathematisch: Nullstellensuche in Intervall auf stetiger Funktion f
- f stetig auf $]a; c[\wedge f(a) < 0 \wedge f(c) > 0 \Rightarrow \exists b, a < b < c : f(b) = 0$
- Hier: „Nullstelle“ ist Übergang zwischen „geht“ und „geht nicht“
- `git bisect` findet garantiert *einen* Übergang
- Aber: es könnte Mehrere geben

Verfahren (beinahe binäre Suche)

- wähle Intervall $]a; c[$, so dass Fehler in c , kein Fehler in a
- wähle b „in der Mitte“ zwischen a und c
- prüfe b auf Fehler
 - Fehler: von vorne mit Intervall a, b
 - kein Fehler: von vorne mit Intervall b, c
- fertig, wenn Intervall leer

- bisect mit HEAD als „kaputt“ markiert und „04711deadbeef“ als „funktionierend“ starten

```
01 git bisect start
02 git bisect bad
03 git bisect good 04711deadbeef
```

- git checkt automatisch Version dazwischen aus, testen und

```
01 git bisect [good|bad|skip]
```

wiederholen bis fertig

- Übersicht

```
01 git bisect log
02 git bisect visualize
```

- n Revisionen im Intervall
- Jeder Schritt halbiert Intervall (ausser bei „skip“)
- Schritte im besten Fall: $\lceil \log_2 n \rceil$
- 10 Schritte für etwa 1000 Revisionen, 20 Schritte für 1 Mio.
- Schritte im schlechtesten Fall: $n - 1$
- Schlechtester Fall, wenn keine Revision baut/testbar ist
- Moral: nur compilierenden Code ins „richtige“ Repository, bzw. in den Hauptentwicklungszweig

spart noch mehr Zeit...

- Nur Teilbäume betrachten

```
01 git bisect start -{}- src/subsystem/subsubsystem
```

- nonplusultra: automagisches `git bisect` – benötigt
 - automatischen Test
 - funktionierendes Build-Skript
 - kleines Skript das beides aufruft

```
01 git bisect run ./test.sh
```

- Moral: automatische Testsuite und funktionierende schnelle Build-Skripte sind toll
siehe gitlab.cs.fau.de/i4/git-bisect-demo.git

Linux Entwicklung

The four essentials freedoms

„The freedom to...

0. ... run the program as you wish, for any purpose.
1. ... study how the program works, and change it so it does your computing as you wish.
2. ... redistribute copies so you can help your neighbor.
3. ... distribute copies of your modified versions to others.

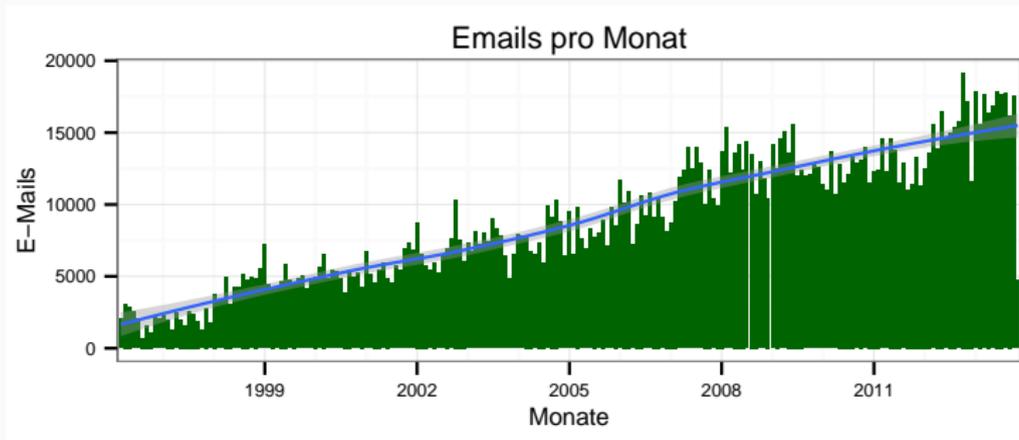
Free/Libre Software and Open-Source Software

- Quelloffene Software bedeutet nicht automatisch freie Software
- Immerwährender Diskurs über die genaue Definition von **freier Software**
 - Beispiel GPL: Stellt Freiheit sicher oder schränkt diese ein?
- “Think free as in free speech, not free beer.” (Gratis versus libre)
- FLOSS

- „The Cathedral and the Bazaar“ (Eric S. Raymond)
 - Essay über Methoden der (Open-Source) Software-Entwicklung
 - Basiert auf Beobachtungen des Entwicklungsprozesses des Linux-Kern
- **Hauptaussagen**
 - „Every good work of software starts by scratching a developer’s personal itch.”
Kapitel 2: „The Mail Must Get Through“
 - „Release early. Release often”
Kapitel 4: „Release Early, Release Often“
 - „If you treat your beta-testers as if they’re your most valuable resource, they will respond by becoming your most valuable resource.”
Kapitel 5: „Is A Rose Not A Rose?“

Linux-Upstream-Entwicklung (Fortsetzung)

- Gesamte **relevante** Kommunikation nur via E-Mail
- Wichtigster Kommunikationskanal:
Linux-Kernel-Mailing-List (LKML)



- Derzeit ca. 7400 Abonnenten, Verbreitung auch über RSS-Feeds, News u.Ä.

Programmcode-Integration

- Source-Code des Linux-Kern
 - Jede Veränderung am Linux-Kern durchläuft den gleichen Prozess
 - Egal ob Fehlerbehebung oder neue Funktionalität
- Kreuzgutachten („Peer-Review“)
 - Gewachsene Hierarchie mit „Benevolent Dictator For Life“
 - Meistens siegt der überlegene Ansatz
- Einflussreichste Aktoren



Linus Torvalds



Andrew Morton



Alan Cox



Greg Kroah-Hartman

Kernel Janitors - Die Hausmeisterei des Linux-Kerns

- Im wesentlichen eine Gruppe von Kernel Entwicklern die kleinere, weniger gravierende Patches entgegen nimmt, sammelt, bewertet und gebündelt in den Entwicklungsprozess einbindet.
- Unterprojekt von <http://kernelnewbies.org>
- Hervorragende, leider leicht angestaubte Anlaufstelle für Neulinge im Linux Entwicklerumfeld

<http://kernelnewbies.org/KernelJanitors/>

Einsenden von Entwicklungen in den Linux-Kern

- Patches im unified diff Format
- Sinnvolle und nachvollziehbare Beschreibungen der Änderungen
- Nur **eine** logische Änderung pro Patch (/Commit)
- Richtige(n) Adressat(en) finden
- Keine Dateianhänge, kein MIME, kein HTML

Einsenden von Entwicklungen in den Linux-Kern (cont.)

- Basisversion genau angeben
- Knappe (70-75 Zeichen!) Kurzbeschreibung, darunter dann ausführliche Beschreibung des Patches und dessen Intention.
- Unnötige Diskussionen über Geschmack (etc.) erkennen, richtig begegnen und vermeiden (↪ siehe bikeshed.org)

Weitere Ressourcen

- Andi Kleen: „On submitting kernel patches”
halobates.de/on-submitting-patches.pdf
- `linux/Documentation/SubmittingPatches`

Auffinden von zuständigen Betreuern

- Änderungen an Dateien im Linux-Kern sind immer an den jeweiligen Betreuer zur Begutachtung einzusenden
- Aufgrund der schieren Größe ist die Zuständigkeit der jeweiligen Dateien aufgeteilt
- Hilfsmittel: `scripts/get_maintainer.pl`:

```
01 $ scripts/get_maintainer.pl -f fs/btrfs/volumes.c
02 Chris Mason <chris.mason@oracle.com>
03 linux-btrfs@vger.kernel.org
04 linux-kernel@vger.kernel.org
```

- Kann auch auf einen Patch direkt angewendet werden, um direkt die betroffenen Dateien und deren Betreuer zu finden.

Linux Coding Style

- Quelltext in Linux werden normalisiert bearbeitet
 - `linux/Documentation/CodingStyle`
 - Im Einzelnen: Einrückung, Lange Zeilen, Setzen von Klammern und nichtdruckbare Zeichen (Whitespace), Bezeichnernamen, Tabulatoren, Kconfig, etc.
 - automatisierter Test: `scripts/checkpatch.pl`
- ⇒ Erleichtert das Lesen und Verständnis, vermeidet Auseinandersetzungen

Linux Coding Style

- Quelltext in Linux werden normalisiert bearbeitet
- `linux/Documentation/CodingStyle`
- Im Einzelnen: Einrückung, Lange Zeilen, Setzen von Klammern und nichtdruckbare Zeichen (Whitespace), Bezeichnernamen, Tabulatoren, Kconfig, etc.
- automatisierter Test: `scripts/checkpatch.pl`

⇒ Erleichtert das Lesen und Verständnis, vermeidet Auseinandersetzungen

Verwendet `checkpatch.pl`!

Verstöße gegen die Richtlinien führen häufig zur Ablehnung des Patches.

Entwicklungen absegnen (1/2)

- In der Linux-Entwicklung wird großer Wert auf korrekte Zuordnung von Patches zu Maintainern gelegt
- Bei einzelnen, kleineren Patches ist häufig unklar, wer der eigentliche Autor ist und was die genauen Nutzungsbedingungen sind
- Per Konvention werden daher Patches von ihren jeweiligen Autoren mit einer speziellen Notation in der Commit-Nachricht abgesegnet:

Signed-off-by: Random J Dev <random@developer.example.org>

Bedeutung

(Auszug, Details in Documentation/SubmittingPatches)

- Der Autor bestätigt das Werk ganz oder in Teilen selbst geschrieben zu haben.
- Der Autor bestätigt das Recht zur Veröffentlichung zu haben
- Der Autor erlaubt die Verwendung und den Vertrieb der Änderung unter den Bedingungen der ursprünglichen Version

- Reported-by: Wer hat das Problem (richtig) gemeldet
- Reviewed-by: Wer hat die Änderung begutachtet
- Tested-by: Wer hat den Patch getestet
- Acked-by: Patch ist zur Kenntnis genommen worden, nicht notwendigerweise aber getestet

Beispiel für Absegnungen

Subject: [PATCH] ACPICA: Fix possible fault in return package object repair code

Fixes a problem that can occur when a lone package object is wrapped with an outer package object in order to conform to the ACPI specification. Can affect these predefined names: _ALR,_MLS,_PSS,_TRT,_TSS,_PRT,_HPX,_DLM,_CSD,_PSD,_TSD

https://bugzilla.kernel.org/show_bug.cgi?id=44171

The bug got introduced by commit 6a99b1c94d053b3420eaa4a4bc in v3.4-rc6, thus it needs to get pushed into 3.4 stable kernels as well.

Reported-by: Vlastimil Babka <caster@gentoo.org>
Tested-by: Vlastimil Babka <caster@gentoo.org>
Tested-by: marc.collin@laboiteaprog.com
Signed-off-by: Bob Moore <robert.moore@intel.com>
Signed-off-by: Lin Ming <ming.m.lin@intel.com>
CC: stable@vger.kernel.org

```
drivers/acpi/acpica/nspredef.c | 2 +-  
1 files changed, 1 insertions(+), 1 deletions(-)
```

„Schöne“ Patchserien bauen mit git

- `git commit --amend`: die vorherige Commit-Nachricht ändern
 - kann ggf. mit `git add` markierte Änderungen hinzufügen
 - z.B. auch mit `--reset-author` oder `--author ...`
- `git checkout abcde -b foo`: einen neuen Seitenzweig „foo“ bei commit abcde auschecken
- `git cherry-pick 12345`: nur Commit „12345“ in aktuellen Zweig übernehmen
- `git rebase -i`: interaktiv eine Serie von Commits editieren
 - ausführliche Beschreibung: <http://git-scm.com/book/en/Git-Tools-Rewriting-History>
- `git add -p`: nur Teile einer Datei für einen Commit auswählen

häufige Probleme und Lösungen

- Patche gegen staging-next oder linux-next, andere Zweige meist die falschen.
- beim Wiedereinreichen von Patchen:
 - Versionszaehler im Betreff
[PATCH v3 2/4] fix something
git format-patch --reroll-count
 - Beschreibung des Patchdeltas zu Vorgängerversion bei diffstat nach ---
git notes
- Signed-off-by:
 - immer alle PASST Gruppenteilnehmer
 - Reihenfolge: Absender zuerst
 - nur funktionierende Email-Adressen, insbesondere nicht xy98abef@fau149man17.cs.fau.de

Beispiel für PATCH v2

From Manjeet Pawar <manjeet.p@samsung.com>
Subject [PATCH v2] ipv6:ipv6_pinfo dereferenced after NULL check
Date Thu, 24 Nov 2016 16:11:57 +0530

From: Rohit Thapliyal <r.thapliyal@samsung.com>

np checked for NULL and then dereferenced. It should be modified for NULL case.

Signed-off-by: Rohit Thapliyal <r.thapliyal@samsung.com>
Signed-off-by: Manjeet Pawar <manjeet.p@samsung.com>
Signed-off-by: Hannes Frederic Sowa <hannes@stressinduktion.org>
Reviewed-by: Akhilesh Kumar <akhilesh.k@samsung.com>

v1->v2: Modified as per the suggestion of Hannes
np ? np->autoflowlabel : ip6_default_np_autolabel(net)

net/ipv6/ip6_output.c | 7 +++++--
1 file changed, 5 insertions(+), 2 deletions(-)

diff --git a/net/ipv6/ip6_output.c b/net/ipv6/ip6_output.c

index 59eb4ed..d734b5e 100644

--- a/net/ipv6/ip6_output.c

+++ b/net/ipv6/ip6_output.c

@@ -215,11 +215,14 @@ int ip6_xmit(const struct sock *sk, struct sk_buff *skb, struct flowi6

Should I just stop attempting to make these trivial fixes?

LKML Post

lkml.org/lkml/2004/12/20/255

Should I just stop attempting to make these trivial cleanups/fixes/whatever patches? are they more noise than gain? am I being a pain to more skilled people on lkml or can you all live with my, sometimes quite ignorant, patches?

Should I just stop attempting to make these trivial fixes?

LKML Post

lkml.org/lkml/2004/12/20/255

Should I just stop attempting to make these trivial cleanups/fixes/whatever patches? are they more noise than gain? am I being a pain to more skilled people on lkml or can you all live with my, sometimes quite ignorant, patches?

- Oftmals wird der Nutzen von kleineren Patches unterschätzt
- Contributor fehlt der Mut den Patch einzureichen
- Und insbesondere **die Ausdauer** den Prozess auch bis zum Ende zu gehen

Should I just stop attempting to make these trivial fixes?

Nicht schüchtern sein

- FLOSS Projekte in der Regel froh über Beiträge
- Allerdings werden die ersten Einreichungen von Neuligen, auch bei scheinbar trivialen Änderungen, in ihrer ursprünglichen Form oftmals **nicht akzeptiert**
- Maintainer sind auch nur Menschen mit unterschiedlichen Persönlichkeiten.

Should I just stop attempting to make these trivial fixes?

Nicht schüchtern sein

- FLOSS Projekte in der Regel froh über Beiträge
 - Allerdings werden die ersten Einreichungen von Neuligen, auch bei scheinbar trivialen Änderungen, in ihrer ursprünglichen Form oftmals **nicht akzeptiert**
 - Maintainer sind auch nur Menschen mit unterschiedlichen Persönlichkeiten.
 - Aber jeder Maintainer auch irgendwo seine persönliche Schmerzgrenze!
-
- Open Source Development and Sustainability: A Look at the Bouncy Castle Project ⇒ [BCCollabTalkSlides.pdf](#)
 - news.ycombinator.com/item?id=16851123

Aufgabe 4

1. Programmcode z.B. des Linux-Staging-Bereich des aktuellen Staging-Baums analysieren und Defekte finden
2. Patches für gefundene Fehler erstellen und ausreichend testen:
 - Patch auf Sourcen des Linux-Kernel anwenden
 - Kompilieren
 - Sicherstellen, dass der betroffene Code übersetzt wurde
`#error ``Attempt to compile this''`
 - Überprüfung des Patches mit `scripts/checkpatch.pl`

Linux fixen (cont.)

3. Patches an `linux-kernel@i4.cs.fau.de` senden, OK abholen
4. Patches an `devel@linuxdriverproject.org` mit Kopie (CC:) an die zuständigen Maintainer (`get_maintainer.pl`) und `linux-kernel@i4.cs.fau.de` senden
5. Vorstellung der Ergebnisse in der Tafelübung (Diskussionsrunde)

Vorstellung der Ergebnisse in der Tafelübung am 21. Juni 2018

Fragen?