Wo sind meine Daten?

René Pfeiffer <pfeiffer@luchs.at>

luchs.at

5. Mai 2010

Inhaltsübersicht - Daten speichern?

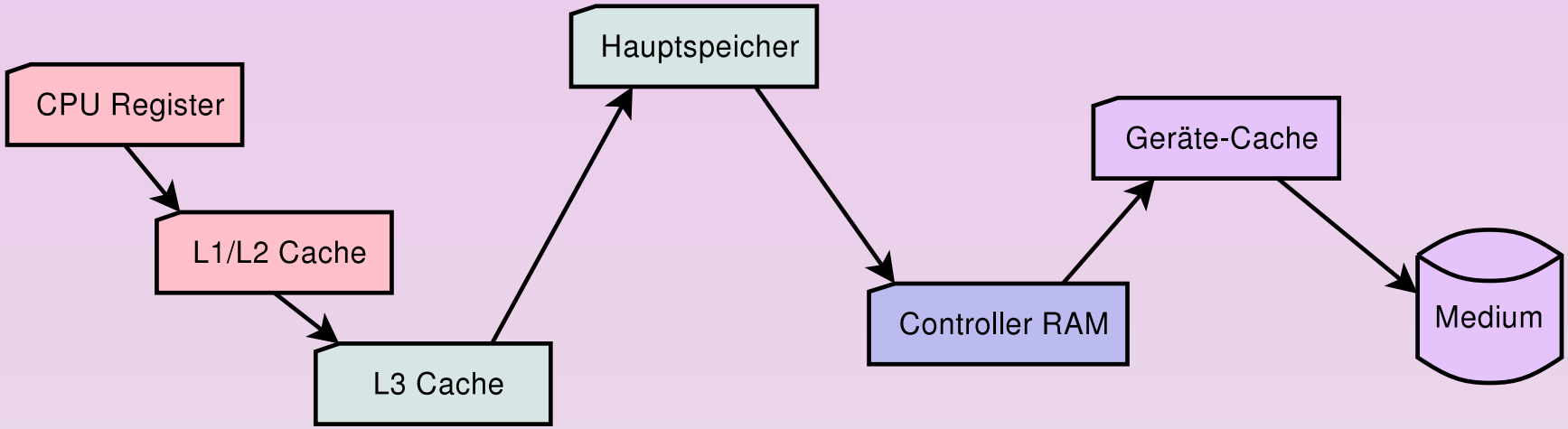
Inhaltsübersicht - Daten speichern?

- Was kann alles schiefgehen?
- Wo liegen Daten?
- Welchen Weg nehmen Daten beim Speichern?
- Welche Schutzmaßnahmen gibt es?
- Was sollten wir uns angewöhnen?

Vorab: Was kann alles schiefgehen?

- Stromausfall / Kurzschluß / Fehlkontakt
- Abstürze - Applikation, Treiber, OS, Firmware
- *Disk full*
- fehlerhafte oder „intelligente” Medien
- Suspend geht, Resume nicht
- Bugs generell und speziell

Weg von Daten durch Hardwareschichten

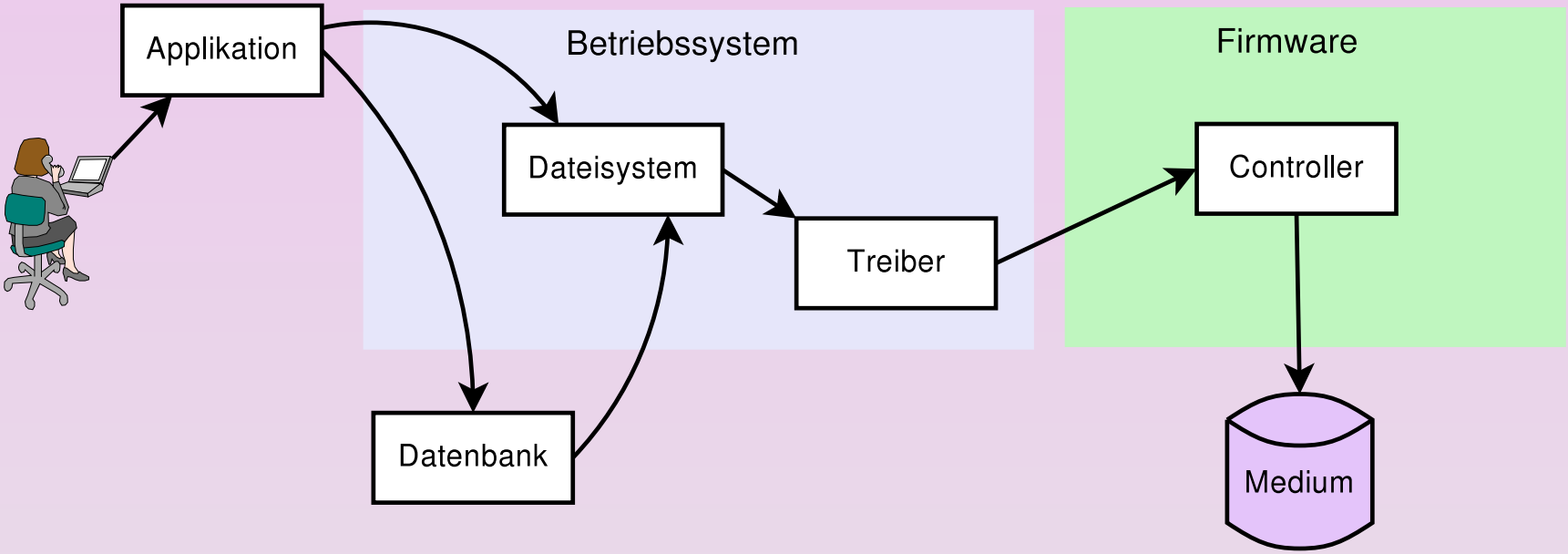


Speicherzugriffszeiten

Komponente	Zeit (s)
Prozessor	$\approx 10^{-9}$
Hauptspeicher	$\approx 10^{-9} - 10^{-7}$
Controller	$\approx 10^{-6} - 10^{-4}$
Festplatten	$\approx 10^{-3}$
Archive	$\approx 10^{-3} - 10^{5}$

Angaben grob in Größenordnungen; Details hängen von Bussystemen und Anordnung der Speicher ab.

Weg von Daten durch Softwareschichten



Wichtige Caches

- Applikationen
  - Speicherverwaltung
- Dateisysteme
- Page Cache (Linuxkern)
  - Daten von Dateien/Blockgeräten
  - Inode/DEntry Cache
  - `mmap()` Dateien
- externe Caches
  - Controller-RAM
  - Gerät-RAM (Festplatten)

Betriebsarten von Caches

Betriebsarten von Caches

- `Read`
  - *Cache Read Miss*
  - *Cache Read Hit*

Betriebsarten von Caches

- `Read`
  - *Cache Read Miss*
  - *Cache Read Hit*
- `Write`
  - *Write-Through* - „Durchschreiben” der Daten
  - *Write-Back* oder *Copy-back* - Verzögern der Schreiboperation

Betriebsarten von Caches

- `Read`
  - *Cache Read Miss*
  - *Cache Read Hit*
- `Write`
  - *Write-Through* - „Durchschreiben” der Daten
  - *Write-Back* oder *Copy-back* - Verzögern der Schreiboperation
- Verzögerungen können im Sekunden- oder Minutenbereich liegen!

Betriebsarten von Caches

- `Read`
  - *Cache Read Miss*
  - *Cache Read Hit*
- `Write`
  - *Write-Through* - „Durchschreiben” der Daten
  - *Write-Back* oder *Copy-back* - Verzögern der Schreiboperation
- Verzögerungen können im Sekunden- oder Minutenbereich liegen!
- Caching kann zu *Out of Order Writes* führen
  - weil Dateisystem *delayed allocation* oder
  - weil Gerät eigenen Cache verwendet

Dateisysteme

Dateisysteme

- Schnittstelle zu Speichermedien
  - Blockgeräte
  - Medien
  - Netzwerke

Dateisysteme

- Schnittstelle zu Speichermedien
  - Blockgeräte
  - Medien
  - Netzwerke
- Organisation von Dateiinformationen (Metadaten)

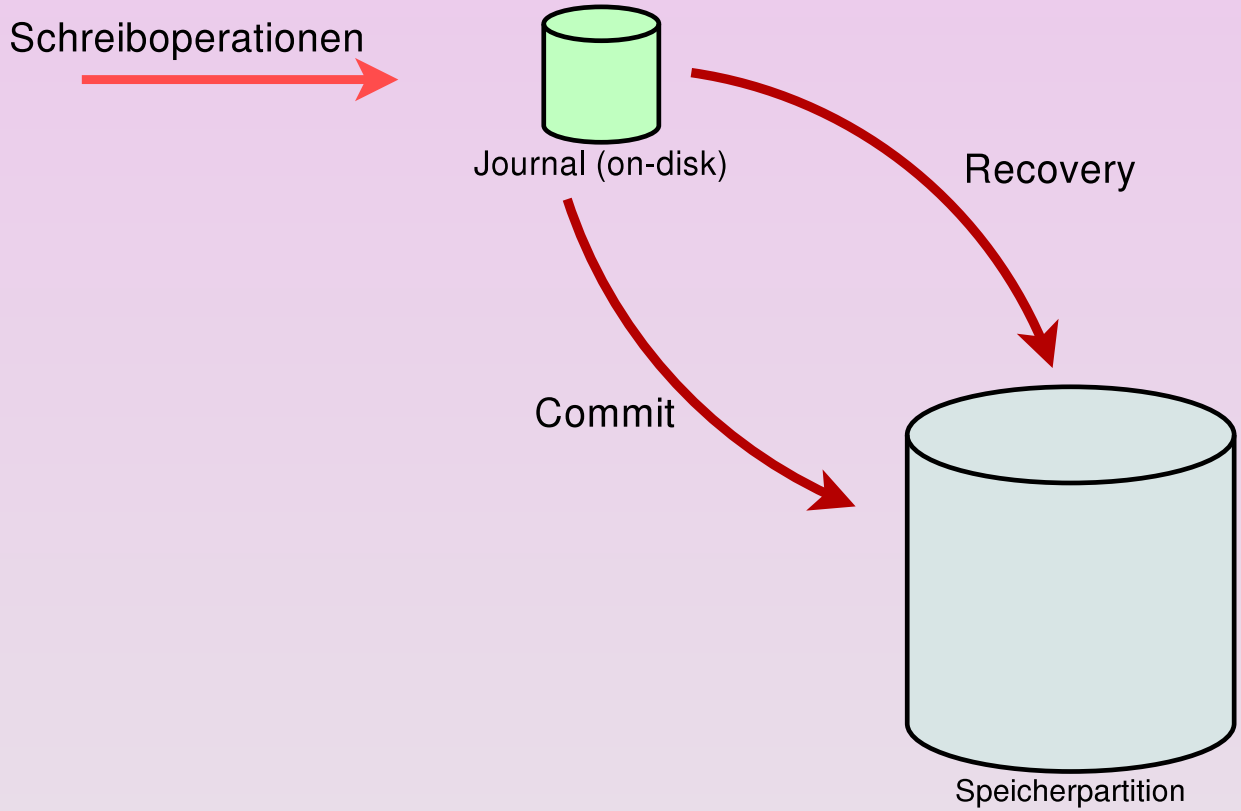
Dateisysteme

- Schnittstelle zu Speichermedien
  - Blockgeräte
  - Medien
  - Netzwerke
- Organisation von Dateiinformationen (Metadaten)
- Verwalten von Dateiinhalten (Daten)

Dateisysteme

- Schnittstelle zu Speichermedien
  - Blockgeräte
  - Medien
  - Netzwerke
- Organisation von Dateiinformationen (Metadaten)
- Verwalten von Dateiinhalten (Daten)
- Speicheroperationen
  - „einfach” (FAT, Ext2, . . .)
  - „Soft Updates” (*BSD)
  - versions-, transaktions-, journal-basiert
  - versionierend

Journaling - Prinzip



Dateisysteme mit Journaling

- Journaling schreibt Änderungen in Log
- Log wird periodisch (zirkulär) geschrieben
- bekannte Vertreter
  - Metadaten & Daten(blöcke)
    - Ext3, Ext4
    - UDF
  - Metadaten
    - JFS
    - NTFS
    - ReiserFS3
    - XFS
  - Daten(blöcke)
    - ReiserFS4
    - ZFS

Liste ist unvollständig.

Journaling Surprise

```
A journaling filesystem exists for one reason only
to provide reasonable behaviour in the event of a
system crash, i.e. to extend the guarantees POSIX
provides and reduce the need to recover data.
```

– xoddam auf LKML

Dateisysteme soll nur **konsistent** sein.

Write Barriers

Write Barriers

- Journaling Dateisysteme müssen bei Commit
  - Vollständigkeit der Transaktionsinformationen sicherstellen,
  - Daten in richtiger Reihenfolge schreiben,
  - Commit Record **zuletzt** schreiben,

Write Barriers

- Journaling Dateisysteme müssen bei Commit
  - Vollständigkeit der Transaktionsinformationen sicherstellen,
  - Daten in richtiger Reihenfolge schreiben,
  - Commit Record **zuletzt** schreiben,

  sonst *Journal kaputt/FS inkonsistent*!!!111!1elf!

Write Barriers

- Journaling Dateisysteme müssen bei Commit
  - Vollständigkeit der Transaktionsinformationen sicherstellen,
  - Daten in richtiger Reihenfolge schreiben,
  - Commit Record **zuletzt** schreiben,

  sonst *Journal kaputt/FS inkonsistent*!!!111!1elf!
- Festplatten haben Caches
- Festplatten ordnen Blöcke zum Schreiben um

Write Barriers

- Journaling Dateisysteme müssen bei Commit
  - Vollständigkeit der Transaktionsinformationen sicherstellen,
  - Daten in richtiger Reihenfolge schreiben,
  - Commit Record **zuletzt** schreiben,

  sonst *Journal kaputt/FS inkonsistent*!!!111!1elf!
- Festplatten haben Caches
- Festplatten ordnen Blöcke zum Schreiben um
- Write Barriers sperren temporär Schreiben von Blöcken
- Sicherung für Commit Records

Write Barriers

- Journaling Dateisysteme müssen bei Commit
  - Vollständigkeit der Transaktionsinformationen sicherstellen,
  - Daten in richtiger Reihenfolge schreiben,
  - Commit Record **zuletzt** schreiben,

  sonst *Journal kaputt/FS inkonsistent*!!!111!1elf!
- Festplatten haben Caches
- Festplatten ordnen Blöcke zum Schreiben um
- Write Barriers sperren temporär Schreiben von Blöcken
- Sicherung für Commit Records
- Barriers kosten Performance ($\approx$ 30%)
- „barriers=0” Default für Ext3/Ext4 („mostly safe”)
- DM/MD: `Disabling barriers, not supported by the underlying device`

Datenbanken

Datenbanken

- Datenbanken haben höhere Anforderungen
  Orphaned money transfers moved to lost+found - unacceptable!

Datenbanken

- Datenbanken haben höhere Anforderungen
  Orphaned money transfers moved to lost+found - unacceptable!
- A.C.I.D.
  - **A**tomicity (Atomarität) - „ganz oder gar nicht”
  - **C**onsistency (Konsistenz) - Bezug auf vor/nach Transaktionen
  - **I**solation - Transaktionen beeinflussen sich nicht
  - **D**urability (Dauerhaftigkeit) - bestätigte Transaktionen werden **dauerhaft** gespeichert

Datenbanken

- Datenbanken haben höhere Anforderungen
  Orphaned money transfers moved to lost+found - unacceptable!
- A.C.I.D.
  - **A**tomicity (Atomarität) - „ganz oder gar nicht”
  - **C**onsistency (Konsistenz) - Bezug auf vor/nach Transaktionen
  - **I**solation - Transaktionen beeinflussen sich nicht
  - **D**urability (Dauerhaftigkeit) - bestätigte Transaktionen werden **dauerhaft** gespeichert
- Datenbanken können Datenblöcke selbst verwalten („raw device support”),

Datenbanken

- Datenbanken haben höhere Anforderungen
  Orphaned money transfers moved to lost+found - unacceptable!
- A.C.I.D.
  - **A**tomicity (Atomarität) - „ganz oder gar nicht”
  - **C**onsistency (Konsistenz) - Bezug auf vor/nach Transaktionen
  - **I**solation - Transaktionen beeinflussen sich nicht
  - **D**urability (Dauerhaftigkeit) - bestätigte Transaktionen werden **dauerhaft** gespeichert
- Datenbanken können Datenblöcke selbst verwalten („raw device support”),
- oft „leben” sie aber auf Dateisystemen.

Datenbanken

- Datenbanken haben höhere Anforderungen
  Orphaned money transfers moved to lost+found - unacceptable!
- A.C.I.D.
  - **A**tomicity (Atomarität) - „ganz oder gar nicht”
  - **C**onsistency (Konsistenz) - Bezug auf vor/nach Transaktionen
  - **I**solation - Transaktionen beeinflussen sich nicht
  - **D**urability (Dauerhaftigkeit) - bestätigte Transaktionen werden **dauerhaft** gespeichert
- Datenbanken können Datenblöcke selbst verwalten („raw device support”),
- oft „leben” sie aber auf Dateisystemen.
- Warum gibt es überhaupt konsistente Datensammlungen in Datenbanken?

fsync()

fsync()

```
A successful close does not guarantee that the
data has been successfully saved to disk, as the
kernel defers writes. It is not common for a file
system to flush the buffers when the stream is
closed. If you need to be sure that the data is
physically stored use fsync(2).
```

– Linux man Page für close()

fsync()

```
A successful close does not guarantee that the
data has been successfully saved to disk, as the
kernel defers writes. It is not common for a file
system to flush the buffers when the stream is
closed. If you need to be sure that the data is
physically stored use fsync(2).
```

– Linux man Page für close()

Stimmt das immer?

fsync()?

```
fsync on Mac OS X: Since on Mac OS X the fsync
command does not make the guarantee that bytes
are written, SQLite sends a F_FULLFSYNC request
to the kernel to ensures that the bytes are
actually written through to the drive platter.
```

Andere Systeme, andere Sitten...

fsync()?

```
fsync on Mac OS X: Since on Mac OS X the fsync
command does not make the guarantee that bytes
are written, SQLite sends a F_FULLFSYNC request
to the kernel to ensures that the bytes are
actually written through to the drive platter.
```

Andere Systeme, andere Sitten...
(ist übrigens POSIX-compliant)
Quelle: Eat my Data

fsync()??

Write Caches der Geräte:

```
- Intel SSD X25E is NOT reliable in default mode.
- To have durability we need to disable write cache
```

– MySQL Performance Blog

fsync()??

Write Caches der Geräte:

```
- Intel SSD X25E is NOT reliable in default mode.
- To have durability we need to disable write cache
```

– MySQL Performance Blog

Problem gibt es auf allen Geräten/Controllern mit Cache, individuelles Speichererlebnis variiert...

aber Storage-Systeme gehen ja nie ungetestet in Produktion...

fsync() im Netzwerk

fsync() im Netzwerk

- TCP/IP benutzt auch Caches - Sendepuffer

fsync() im Netzwerk

- TCP/IP benutzt auch Caches - Sendepuffer
- TCP PUSH - eine Art `fsync()`
  - *Asks to push the buffered data to the receiving application.*
  - MASQUERADE, Time Outs, Link Down, ...

fsync() im Netzwerk

- TCP/IP benutzt auch Caches - Sendepuffer
- TCP PUSH - eine Art `fsync()`
  - *Asks to push the buffered data to the receiving application.*
  - MASQUERADE, Time Outs, Link Down, ...
- HTTP POST ist Schreiboperation

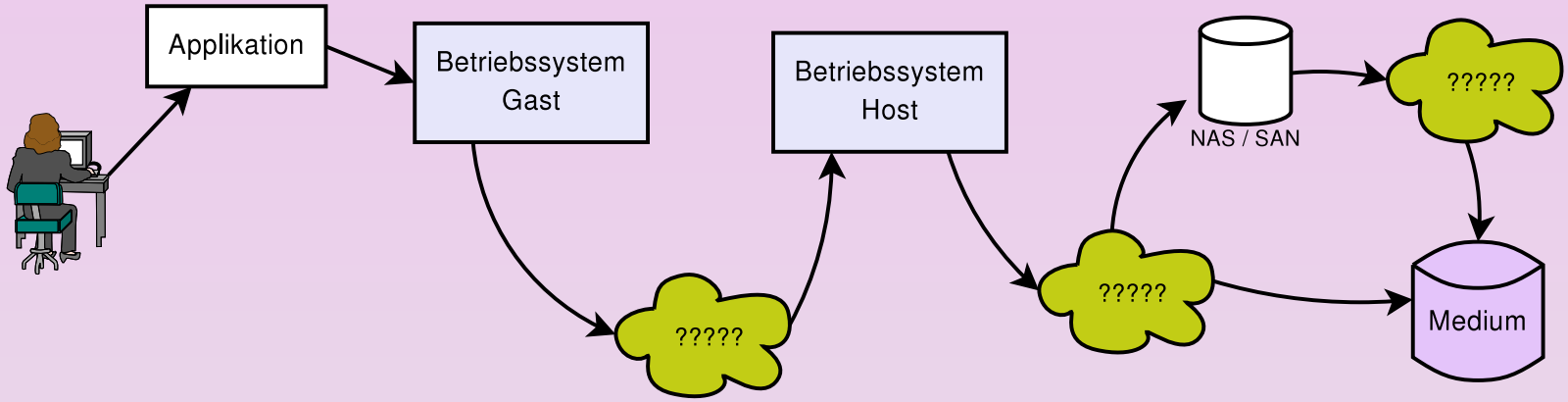
fsync() im Netzwerk

- TCP/IP benutzt auch Caches - Sendepuffer
- TCP PUSH - eine Art `fsync()`
  - *Asks to push the buffered data to the receiving application.*
  - MASQUERADE, Time Outs, Link Down, ...
- HTTP POST ist Schreiboperation
- The NFS protocol(s) don't know anything about fsync(2). It is up the individual client implementation to support fsync(2) or not, depending upon its own set of semantics.

fsync() im Netzwerk

- TCP/IP benutzt auch Caches - Sendepuffer
- TCP PUSH - eine Art `fsync()`
  - *Asks to push the buffered data to the receiving application.*
  - MASQUERADE, Time Outs, Link Down, ...
- HTTP POST ist Schreiboperation
- The NFS protocol(s) don't know anything about fsync(2). It is up the individual client implementation to support fsync(2) or not, depending upon its own set of semantics.
- NFS hat Caches: *NFS caching is a big problem when multiple computers are accessing the same mailbox simultaneously. The best fix for this is to prevent it from happening.*

Weg von Daten durch Virtualisierung



Weg der Daten durch Virtualisierung (2)

Weg der Daten durch Virtualisierung (2)

- Gastsysteme haben
  - Applikationen
  - Dateisysteme
  - Treiber & Blockgeräte

Weg der Daten durch Virtualisierung (2)

- Gastsysteme haben
  - Applikationen
  - Dateisysteme
  - Treiber & Blockgeräte
- Hostsysteme haben
  - Applikationen
  - Dateisysteme
  - Treiber & Blockgeräte

Weg der Daten durch Virtualisierung (2)

- Gastsysteme haben
  - Applikationen
  - Dateisysteme
  - Treiber & Blockgeräte
- Hostsysteme haben
  - Applikationen
  - Dateisysteme
  - Treiber & Blockgeräte
- Verschachtelung++
  - FS auf FS (auf FS, . . .)
  - Cache Flushes im Überfluß
  - mehrfaches Write Reordering / Delayed Allocation
  - Spaß am Gerät - rote/blaue Pillen inbegriffen

Gibt es Hoffnung?

Wege aus der Hölle (Systemadministration)

- Auswählen
  - Dateisysteme & (Mount) Optionen
  - OS Parameter
  - Controller, Speicher mit ECC
  - RAID-Level, LVMs
  - Speichermedien
  - **Notstrom! Batterien!**
- ¡Testen!
  - Lights out! - unkontrolliertes Herunterfahren
  - Defekte unter Last - Kabel ziehen
  - App/Treiber/OS darf nicht abstürzen
- Sichern!
  - Backups
  - Archive

Wege aus der Hölle (Entwickler)

- Plattform
  - Plattform = Libs + Kern + OS
  - Eigenheiten/Bugs kennen
  - Möglichkeiten kennen & nutzen
- Qualität
  - undefinierte Zustände vermeiden
  - alle Fehler abfangen
  - **nicht** auf Dateisystem verlassen!
  - temporäres File schreiben, bei Erfolg umbenennen
  - **fflush()**, **fsync()**, ...

Beispiel aus Eat My Data Vortrag:

```
FILE *f;
f= fopen("important_document.temp","w");
if(!f) return errno;
size_t w= fwrite(d.document,d.len,1,f);
if(w<d.len) return errno;
if(fflush(f)!=0) return errno;
if(fsync(fileno(f))==-1) return errno;
fclose(f);
rename("important_document.temp","important_document");
```

Beispiel aus Eat My Data Vortrag:

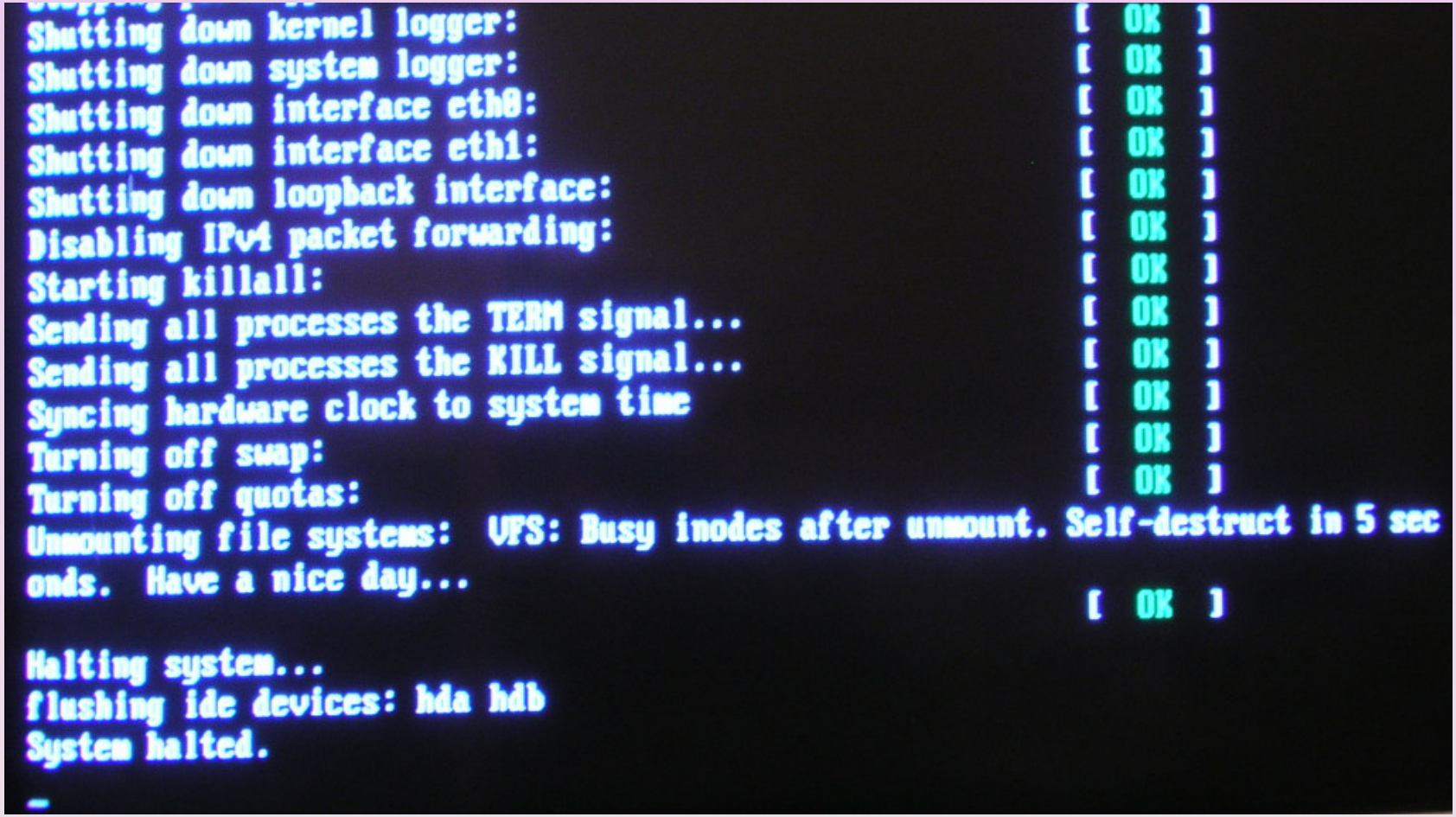
```
FILE *f;
f= fopen("important_document.temp","w");
if(!f) return errno;
size_t w= fwrite(d.document,d.len,1,f);
if(w<d.len) return errno;
if(fflush(f)!=0) return errno;
if(fsync(fileno(f))==-1) return errno;
fclose(f);
rename("important_document.temp","important_document");
```

Funktioniert übrigens nicht gut mit **großen** Dateien, die oft verändert werden.

Wege aus der Hölle (Benutzer)

- Applikation / System
  - Eigentheiten/Bugs kennen
  - Möglichkeiten kennen & nutzen
- Gewohnheiten
  - kein blindes Vertrauen
  - oft (genug) Speichern & Kopieren
  - Backups
  - Archive
- Beschweren
  - Bug Reports
  - Wish Lists

Noch Fragen?


Shutting down kernel logger: [OK]
Shutting down system logger: [OK]
Shutting down interface eth0: [OK]
Shutting down interface eth1: [OK]
Shutting down loopback interface: [OK]
Disabling IPv4 packet forwarding: [OK]
Starting killall: [OK]
Sending all processes the TERM signal... [OK]
Sending all processes the KILL signal... [OK]
Syncing hardware clock to system time [OK]
Turning off swap: [OK]
Turning off quotas: [OK]
Unmounting file systems: VFS: Busy inodes after unmount. Self-destruct in 5 seconds. Have a nice day...
[OK]
Halting system...
Flushing ide devices: hda hdb
System halted.

Über dieses Dokument

- Autor: René Pfeiffer
- Erstellt mit LaTeX und LaTeX Beamer Class
- Dokumentensammlung unter

  http://web.luchs.at/information/docs.php