

# *FUSE: Filesystem in Userspace*

3. Juli 2007

Seminar Dateisysteme

Sebastian Sproesser

sproess@mathi.uni-heidelberg.de



# *FUSE: Filesystem in Userspace*

## *Agenda*

- Was ist FUSE?
    - Grundsätzliches
    - Historie
  - Warum FUSE?
    - Vor-/Nachteile
  - Wie funktioniert FUSE?
    - Technische Details
    - Beispielprogramm
  - Literatur / Quellen
  - Beispielanwendungen
- 
-

# *Was ist FUSE?*

## *Grundsätzliches*

Formelle Definition von “Dateisystem”:

Eine Menge von abstrakte Datentypen für

- Speicherung
- hierarchische Organisation
- Manipulation
- Navigation
- Zugriff
- Wiederfinden

von Daten.

---

---

# *Was ist FUSE?*

## *Grundsätzliches*

“Normale” Dateisysteme Spezialfall

- Weniger: “Wie schreibe ich konkrete Daten auf einen Festspeicher”
  - Öfters: “Wie kann ich Daten in eine Dateien/Ordner-Struktur abbilden”
  - Ausnahmen bestätigen die Regel
- 
-

# *Was ist FUSE*

## *Historie*

- Ursprünglich Teil von “A Virtual Filesystem” (AVFS)
- Seit 2004 eigenständiges Projekt
- Offiziell im Linux-Kernel seit 2.6.14 (Oktober 2005)

# *Warum FUSE?*

## *Vorteile (I)*

- Einfache API
- Sichere Implementation
- Userspace-Kernel-Interface effizient
- Nicht-Privilegierte Benutzer
- Kernel bleibt schlanker
- Lizenzkonflikte umgehen

# *Warum FUSE?*

## *Vorteile (II)*

- Sehr stabil
  - Unterstützte OS: linux-2.4.X, linux 2.6.X, FreeBSD, OpenSolaris, Mac OS X, NetBSD, Windows?
  - Language Bindings: C, C++, Java, C#, Haskell, Tcl, Python, Perl, Sh (!), SWIG, Ocaml, Pliant, Ruby
- 
-

# *Warum FUSE?*

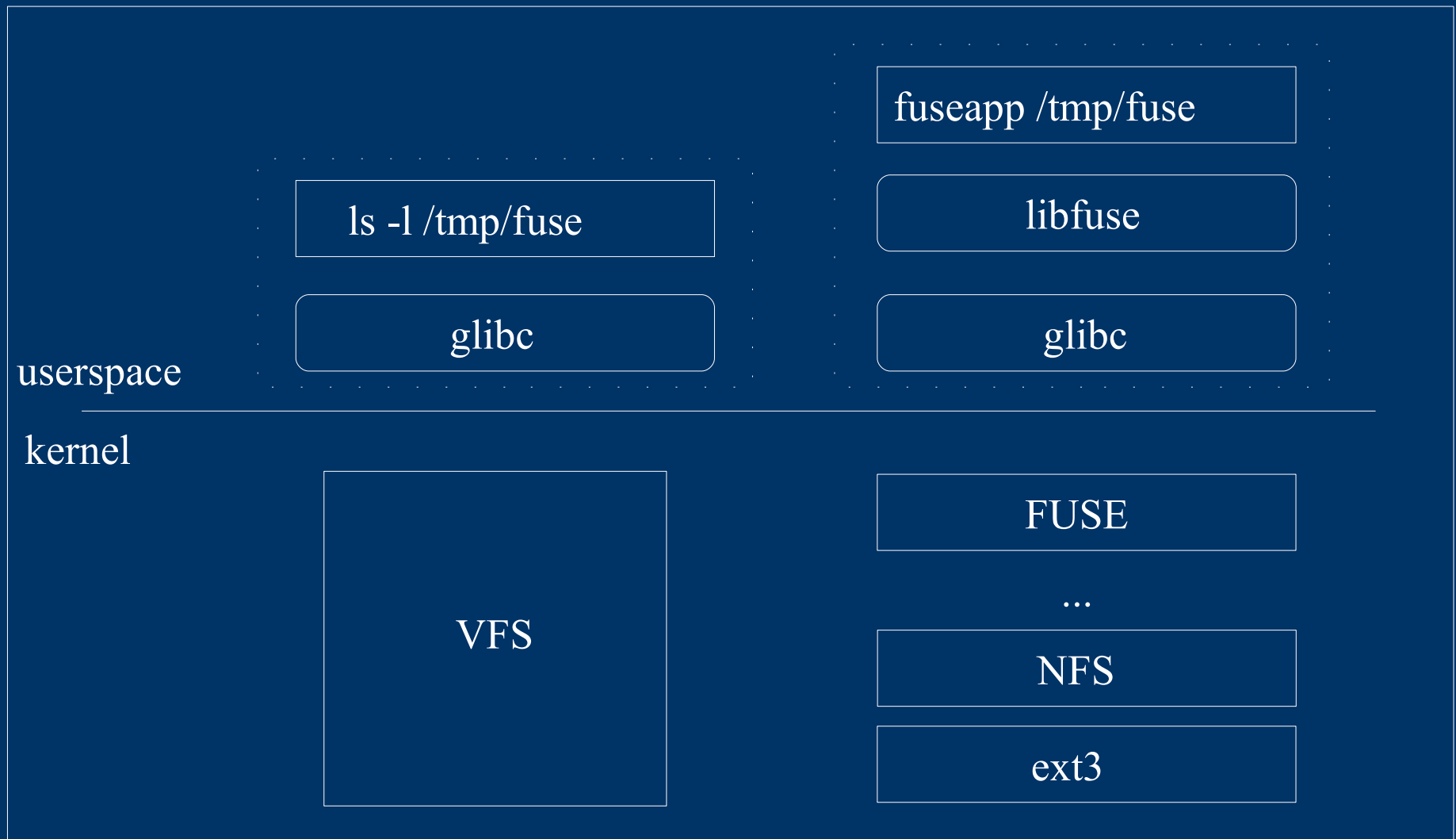
## *Nachteile*

- Trotz allem eine (zusätzliche) Schicht
- FUSE zwar sicher, aber auch die Anwendungen?
- (Wenig Perlen, viel Müll)



# Wie funktioniert FUSE?

## Technische Details



# Wie funktioniert FUSE?

## Beispielprogramm "Hello World"-FS

```
#include <fuse.h>
[...]
```

```
static int hello_getattr(const char *path, [...]);
static int hello_readdir(const char *path, [...]);
static int hello_open(const char *path, [...]);
static int hello_read(const char *path, [...]);
```

```
static struct fuse_operations hello_oper = {
    .getattr    = hello_getattr,
    .readdir    = hello_readdir,
    .open       = hello_open,
    .read       = hello_read,
};
```

```
int main(int argc, char *argv[]) {
    return fuse_main(argc, argv, &hello_oper);
}
```

---

---

# *Wie funktioniert FUSE?*

## *Beispielprogramm "Hello World"-FS*

- Livedemo -

---

---

# *Literatur / Quellen*

- [http://en.wikipedia.org/wiki/Filesystem\\_in\\_Userspace](http://en.wikipedia.org/wiki/Filesystem_in_Userspace)
  - <http://fuse.sourceforge.net/>
  - Callback File System: <http://www.eldos.com/cbfs/>
- 
-

# *Beispielanwendungen*

## *GlusterFS*

- Geclustertes Dateisystem
  - Mehrere “Bricks” zu parallelem Netz-FS
  - Skaliert bis zu mehreren Petabyte
  - Unterstützt Infiniband, TCP/IP
  - Sehr performant (Design, Parallelisierung, Multicast)
  - Offene Architektur (Scheduler, Translator)
  - Kein Single Point of Failure
- 
-

# *Beispielanwendungen*

## *GlusterFS*

- GlusterHPC für “High Performance Computing”
- GlusterEP für einfache Replikation
- <http://www.gluster.org/>

# *Beispielanwendungen*

## *SSHFS*

- Entferntes Verzeichnis transparent in lokalen Baum einbinden
- SFTP (SSH-Subsystem)
- `sshfs user@host:/path/ /mountpoint`

# *Beispielanwendungen*

## *CurlFtpFS*

- Analog zu SSHFS
- FTP-Verzeichnisse transparent einbinden
- `curlftpfs ftp://host/ mountpoint`

# *Beispielanwendungen*

## *Wayback*

- FUSEs Antwort (eher Frage) auf Apples Time Machine
  - CVS auf FS- und Transaktions-Basis
  - Transparente “Versionierung” auch für bestehende Verzeichnisse
  - Leider seit 2004 kein Update
- 
-

# *Beispielanwendungen*

## *httpFS / fuseiso*

- httpFS: Web-Dateien byteweise
- fuseiso: ISO-Dateien als Benutzer
- In Kombination sehr mächtig

# *Beispielanwendungen*

## *GmailFS*

- gmail-Account als Speicherplatz
  - Momentan knapp 3 GB
  - Dateien als (segmentierte) Attachments
  - Unverschlüsselt
  - Legal nach terms of use?
- 
-

# *Beispielanwendungen*

## *ClamFS*

- Transparenter Virencheck durch ClamAV
- Caching-Mechanismus
- Bei Virus: Dateizugriff blockiert, Mail an Administrator

# *Beispielanwendungen*

## *WikipediaFS*

- Wikipedia- (oder allgemein Mediawiki-)Artikel im Texteditor bearbeiten
- WikipediaFS stellt den http-Layer
- Navigieren nicht trivial (kein brauchbares `ls`)
- Eins von vielen FUSE-Webzwonull-Interfaces (FlickrFS, BlogFS, DuggFS, ...)

# *Beispielanwendungen*

## *tagofs*

- Musiksammlung (MP3, OGG) anhand von Tags durchstöbern
- Eins von vielen “Dateien/Daten aufgrund von Metadaten neu anordnen”-Dateisystemen (RelFS, MythTVFS, playlistfs, ...)

# *Beispielanwendungen*

## *Bluetooth File System*

- Dateien per OBEX von und zu Bluetooth-Geräten transferieren
- Eins von vielen “Hardware geschickt in Verzeichnisbaum abbilden”-Dateisystemen (DVDfs, Sief, FUSEPod, gphoto2-fuse-fs, ...)

# *FUSE: Filesystem in Userspace*

3. Juli 2007  
Seminar Dateisysteme

Sebastian Sproesser  
sproess@mathi.uni-heidelberg.de



## ***FUSE: Filesystem in Userspace***

### ***Agenda***

- Was ist FUSE?
    - Grundsätzliches
    - Historie
  - Warum FUSE?
    - Vor-/Nachteile
  - Wie funktioniert FUSE?
    - Technische Details
    - Beispielprogramm
  - Literatur / Quellen
  - Beispielanwendungen
- 
-

## **Was ist FUSE?** **Grundsätzliches**

Formelle Definition von “Dateisystem”:

Eine Menge von abstrakte Datentypen für

- Speicherung
- hierarchische Organisation
- Manipulation
- Navigation
- Zugriff
- Wiederfinden

von Daten.

---

---

Definition von [http://en.wikipedia.org/wiki/File\\_system](http://en.wikipedia.org/wiki/File_system)

## **Was ist FUSE?** **Grundsätzliches**

“Normale” Dateisysteme Spezialfall

- Weniger: “Wie schreibe ich konkrete Daten auf einen Festspeicher”
- Öfters: “Wie kann ich Daten in eine Dateien/Ordner-Struktur abbilden”
- Ausnahmen bestätigen die Regel

•FUSE hält sich an die formelle Definition, damit sind sowohl traditionelle, als auch abstrakte Dateisysteme möglich.

Beispiel für traditionelle Dateisysteme:

- NTFS-3G
- ext2fuse

Beispiel für abstrakte Dateisysteme:

- GMailFS
- WikipediaFS
- btfs

Mischung:

- EncFS
- SSHFS

## **Was ist FUSE**

### **Historie**

- Ursprünglich Teil von “A Virtual Filesystem” (AVFS)
- Seit 2004 eigenständiges Projekt
- Offiziell im Linux-Kernel seit 2.6.14 (Oktober 2005)

NetBSD hat eigenen userspace-FS-Mechanismus PUFFS (Pass-to-Userspace Framework File System) mit FUSE-Kompatibilitätslayer

Windows-Port in Entwicklung?

Ähnlicher Windows-Mechanismus: “Callback File System”

## Warum FUSE? Vorteile (I)

- Einfache API
- Sichere Implementation
- Userspace-Kernel-Interface effizient
- Nicht-Privilegierte Benutzer
- Kernel bleibt schlanker
- Lizenzkonflikte umgehen

Einfache API:

“Hello World”-FS in <100 Zeilen

Komplexere Software auch in wenig Zeilen

httpfs: <1000 Zeilen

fuseiso: ~2000 Zeilen

Sichere Implementation, effizientes Interface:

```
`cat /usr/src/linux/fs/fuse/*.c | sort | uniq | grep -c .` => 2117
```

Nicht-Privilegierte Benutzer:

vermutlich größtes Pro-Argument

Lizenzkonflikte umgehen:

ermöglicht ZFS für Linux

## **Warum FUSE?**

### **Vorteile (II)**

- Sehr stabil
- Unterstützte OS: linux-2.4.X, linux 2.6.X, FreeBSD, OpenSolaris, Mac OS X, NetBSD, Windows?
- Language Bindings: C, C++, Java, C#, Haskell, Tcl, Python, Perl, Sh (!), SWIG, Ocaml, Pliant, Ruby

Stabil: Seit fast 2 Jahren ohne größere Probleme im Kernel

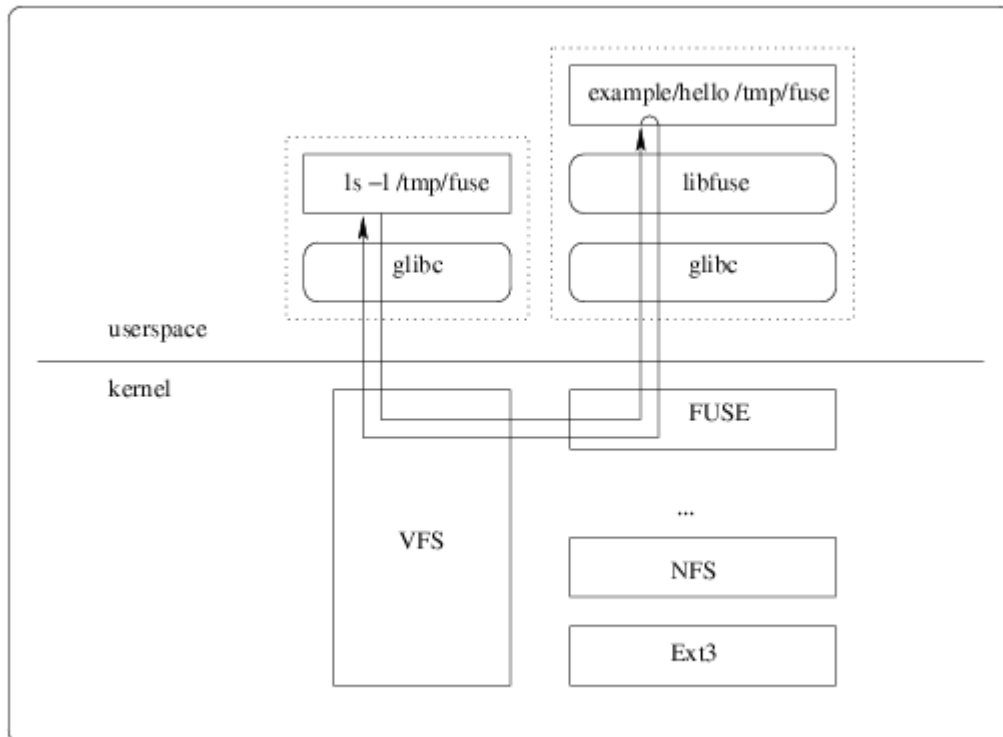
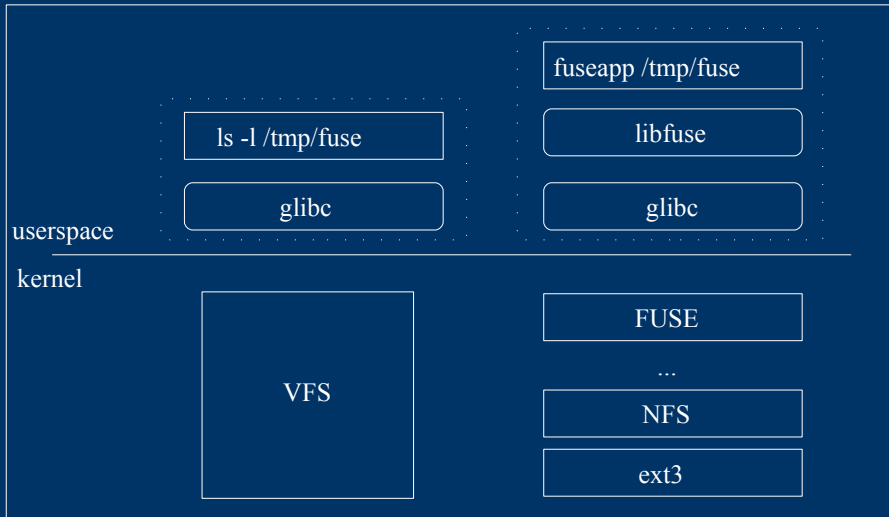
SWIG: “Simplified Wrapper and Interface Generator” => Tcl, Perl, Python, Ruby, PHP, Lua, Java, C#, Scheme, Ocaml

## **Warum FUSE?** **Nachteile**

- Trotz allem eine (zusätzliche) Schicht
- FUSE zwar sicher, aber auch die Anwendungen?
- (Wenig Perlen, viel Müll)



# Wie funktioniert FUSE? Technische Details



## Wie funktioniert FUSE? Beispielprogramm "Hello World"-FS

```
#include <fuse.h>
[...]  
static int hello_getattr(const char *path, [...]);  
static int hello_readdir(const char *path, [...]);  
static int hello_open(const char *path, [...]);  
static int hello_read(const char *path, [...]);  
  
static struct fuse_operations hello_oper = {  
    .getattr = hello_getattr,  
    .readdir = hello_readdir,  
    .open    = hello_open,  
    .read    = hello_read,  
};  
  
int main(int argc, char *argv[]) {  
    return fuse_main(argc, argv, &hello_oper);  
}
```

Komplette Quellen: <http://fuse.sourceforge.net/helloworld.html>

getattr, readdir, open, read, ... => POSIX-Dateisystem-Operationen

## **Wie funktioniert FUSE?**

### **Beispielprogramm "Hello World"-FS**

- Livedemo -

Livedemo:

Quelltext des Hello-World-FS.

```
~/fuse/example$ mkdir /tmp/fuse
~/fuse/example$ ./hello /tmp/fuse
~/fuse/example$ ls -l /tmp/fuse
total 0
-r--r--r--          1   root root 13   Jan  1   1970
  hello
~/fuse/example$ cat /tmp/fuse/hello
Hello World!
~/fuse/example$ fusermount -u /tmp/fuse
~/fuse/example$
```

## *Literatur / Quellen*

- [http://en.wikipedia.org/wiki/Filesystem\\_in\\_Userspace](http://en.wikipedia.org/wiki/Filesystem_in_Userspace)
  - <http://fuse.sourceforge.net/>
  - Callback File System: <http://www.eldos.com/cbfs/>
- 
-

## **Beispielanwendungen**

### **GlusterFS**

- Geclustertes Dateisystem
- Mehrere “Bricks” zu parallelem Netz-FS
- Skaliert bis zu mehreren Petabyte
- Unterstützt Infiniband, TCP/IP
- Sehr performant (Design, Parallelisierung, Multicast)
- Offene Architektur (Scheduler, Translator)
- Kein Single Point of Failure

- Entwickler raten von Striping ab, das darunterliegende Dateisystem kann das besser
- Scheduler:
  - \* NUFA (Non-Uniform FS Scheduler): Lokales System höhere Priorität als Remote-Systeme
  - \* ALU (Adaptive Least Usage): Verschiedene Indikatoren (HDD-Auslastung, geöffnete Dateien, HDD-Geschwindigkeit, ...)
  - \* random
  - \* rr (round robin)
- Translator: Mechanismus aus Hurd, FS-Fähigkeiten durch definiertes Interface erweiterbar, z.B.:
  - \* stripe
  - \* prefetch
  - \* encryption

## ***Beispielanwendungen*** ***GlusterFS***

- GlusterHPC für “High Performance Computing”
  - GlusterEP für einfache Replikation
  - <http://www.gluster.org/>
- 
-

## **Beispielanwendungen** **SSHFS**

- Entferntes Verzeichnis transparent in lokalen Baum einbinden
  - SFTP (SSH-Subsystem)
  - `sshfs user@host:/path/ /mountpoint`
- 
-

## *Beispielanwendungen*

### *CurlFtpFS*

- Analog zu SSHFS
- FTP-Verzeichnisse transparent einbinden
- `curlftpfs ftp://host/ mountpoint`

## ***Beispielanwendungen*** ***Wayback***

- FUSEs Antwort (eher Frage) auf Apples Time Machine
  - CVS auf FS- und Transaktions-Basis
  - Transparente “Versionierung” auch für bestehende Verzeichnisse
  - Leider seit 2004 kein Update
- 
-

## *Beispielanwendungen*

### *httpFS / fuseiso*

- httpFS: Web-Dateien byteweise
- fuseiso: ISO-Dateien als Benutzer
- In Kombination sehr mächtig

```
user@host $ httpfs http://ftp.rz.tu-  
bs.de/pub/mirror/knoppix/knoppix-  
dvd/KNOPPIX_V5.1.1DVD-2007-01-04-EN.iso mountpoint1  
user@host $ fuseiso  
mountpoint1/KNOPPIX_V5.1.1DVD-2007-01-04-EN.iso  
mountpoint2  
user@host $ ls mountpoint2  
KNOPPIX Software autorun.bat autorun.inf autorun.pif  
books boot cdrom.ico index.html qemu-0.8.2
```

## **Beispielanwendungen**

### **GmailFS**

- gmail-Account als Speicherplatz
- Momentan knapp 3 GB
- Dateien als (segmentierte) Attachments
- Unverschlüsselt
- Legal nach terms of use?

filesystem-level encryption zur Verschlüsselung

Keine direkte Verletzung der terms of use. ABER: "Google reserves the right to refuse service to anyone at any time without notice for any reason."

## ***Beispielanwendungen*** ***ClamFS***

- Transparenter Virencheck durch ClamAV
  - Caching-Mechanismus
  - Bei Virus: Dateizugriff blockiert, Mail an Administrator
- 
-

## **Beispielanwendungen**

### **WikipediaFS**

- Wikipedia- (oder allgemein Mediawiki-)Artikel im Texteditor bearbeiten
- WikipediaFS stellt den http-Layer
- Navigieren nicht trivial (kein brauchbares ls)
- Eins von vielen FUSE-Webzwoonull-Interfaces (FlickrFS, BlogFS, DuggFS, ...)

- FlickrFS: Zugriff, Down- und Upload auf Flickr-Account
- BlogFS: Bloggen per Kommandozeile
- DuggFS: DIGG-Interface

## **Beispielanwendungen** **tagsfs**

- Musiksammlung (MP3, OGG) anhand von Tags durchstöbern
- Eins von vielen “Dateien/Daten aufgrund von Metadaten neu anordnen”-Dateisystemen (RelFS, MythTVFS, playlistfs, ...)

- RelFS: Zugriff auf relationales DBMS per Dateibauem, Views als Verzeichnisse, ...
- MythTVFS: Sendungen aufgrund Name, Datum, Genre, ... ordnen
- playlistfs: Musik-Playlists (z.B. .m3u) als Ordner darstellen

## **Beispielanwendungen** **Bluetooth File System**

- Dateien per OBEX von und zu Bluetooth-Geräten transferieren
- Eins von vielen “Hardware geschickt in Verzeichnisbaum abbilden”-Dateisystemen (DVDfs, SiefS, FUSEPod, gphoto2-fuse-fs, ...)

- DVDfs: Video-DVDs on-the-fly entschlüsseln
- SiefS: Zugriff auf Siemens-Mobiltelefone
- FUSEPod: iPod-Interface
- gphoto2-fuse-fs: Digitalkameras