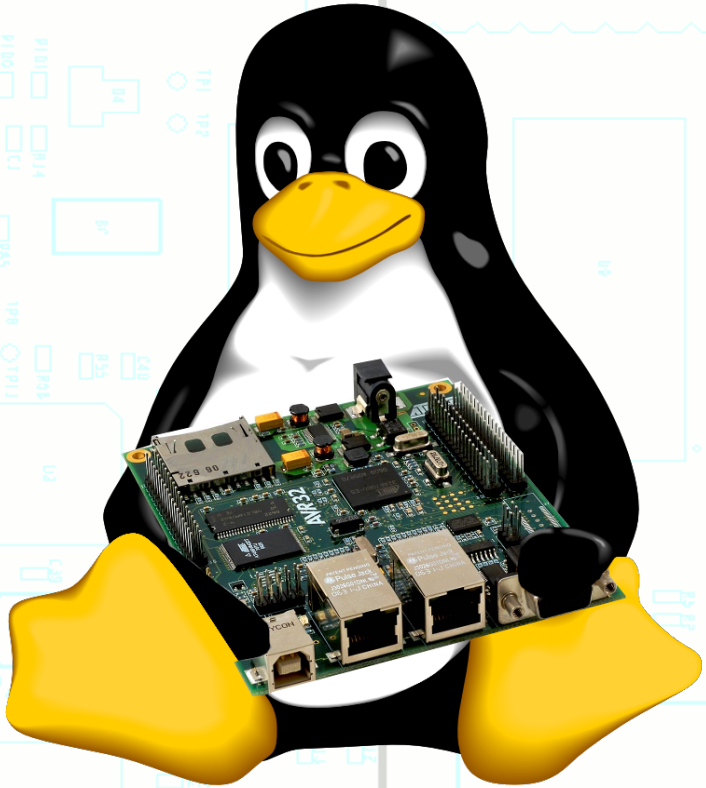


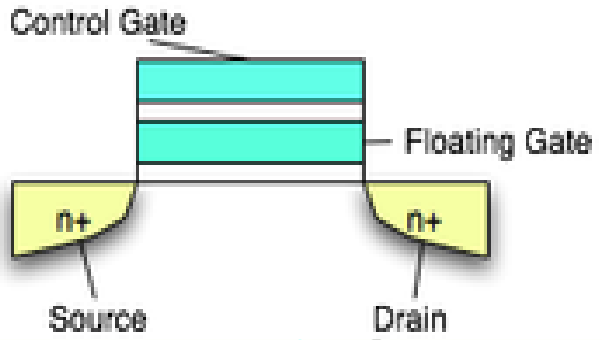
Speichertechnologien & Filesysteme

Agenda

- **Speichertechnologien**
  - Flash Speicher
  - MTD (Memory Technology Devices)
- **Filesysteme**
  - RAMFS
  - CRAMFS
  - JFFS2
  - UBIFS
  - LOGFS
  - YAFFS

Flash Speicher Entwicklung

- **FGMOS:**
  - Bell Laboratories
  - 1967 von D. Kahng und S.M. Sze
  - Feldeffekttransistor mit isoliertem Gate
  - Speicherung der Ladung im Floating Gate
  - Grundlage aller nichtflüchtigen Speicher



Flash Speicher Entwicklung

- **EPROM:**
  - Erasable Programmable read-only Memory
  - Matrix aus Speicherzellen
  - (MOSFET – Transistor + Floating Gate)
  - 1 Bit := Ein Transistor
  - Schreiben mit einem „EPROM“ Brenner
  - Löschen mit UV-Licht

Flash Speicher Entwicklung

- **EEPROM:**
  - electrically erasable programmable read-only memory,
  - Weiterentwicklung des EPROM
  - Löschen nun ohne UV-Licht möglich erheblich schneller als EPROM
  - Schreiben + Löschen direkt im Gerät
  - Byte-weise Ansteuerung möglich
  - Schreibzyklus 1ms - 10ms

Flash Speicher

- **Flash – EEPROM:**
  - Löschen nur in Blöcken möglich
  - Schreibzyklus 1us – 1ms !
  - Weiterentwicklung des EEPROM
- **NAND**
  - Serienschaltung von Speicherzellen
  - Eine Datenleitung
  - Viel Speicher auf wenig Platz
  - Aber: hohe Zugriffszeiten auf einzelnes Datum
- **NOR**
  - Parallelschaltung
  - Geringere Zugriffszeit
  - Einsatz als Programmspeicher von Mikrocontrollern

Flash Speicher

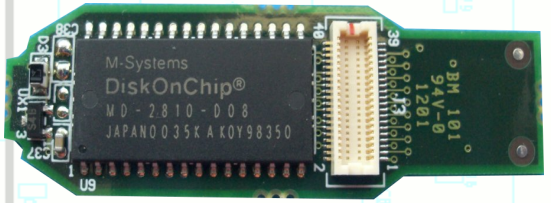
- 1984: Entwicklung von Toshiba
- 1994: Erstes CFM von SanDisk (4MB)
- 1998: Erster Memory Stick von Sony

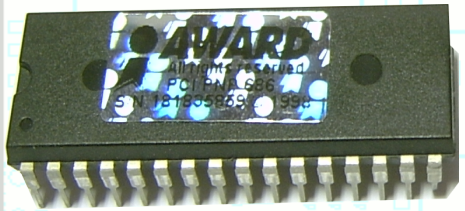
- **Heute:**
  - Speicherkarten (Digitalkamera, Handy, Handheld)
  - USB-Sticks
  - MP3-Player
  - Mikrocontroller (embedded flash)

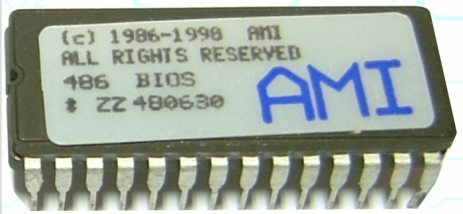
- **Zukunft:**
  - Bei Computern als Ersatz/Unterstützung von Festplatten (vgl. Hybridfestplatten)

Flash Speicher

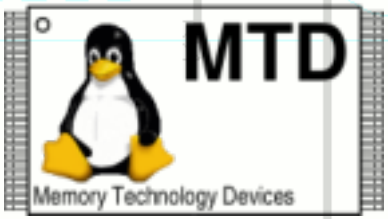
- USB-Stick
- SD-Card
- Flash-EEPROM-IC
- Flash-EPROM-IC





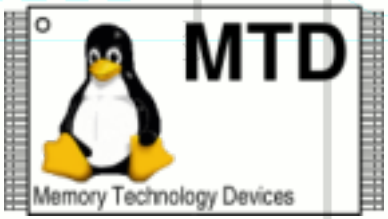


MTD
(Memory Technology Devices)



- generisches Linux Teilsystem für Speicher-Geräte
- **Ziel von MTD:**
  Durch Angebot einer generischen Schnittstelle zwischen den Hardware-Treibern und dem „*upper layer*“ des Systems soll das Schreiben von Treibern für neue Hardware erleichtert werden.
- Die Treiber müssen nichts über das Speicher-format des Geräts wissen und müssen lediglich einfache Routinen wie *lesen*, *schreiben* und *löschen* zur Verfügung stellen.

MTD
(Memory Technology Devices)



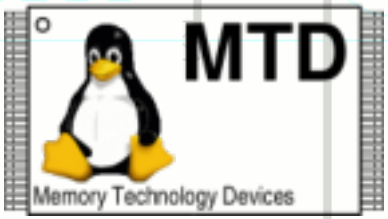
- **UNIX kennt zwei Arten von Geräten**:
  - Character Devices:
    - können nicht durchsucht werden
    - haben keine feste Größe

  - Block Devices:
    - können durchsucht werden
    - haben eine feste Größe.
    - werden in Blöcke mit mehreren Bytes aufgeteilt (im Normalfall 512).

- Flash passt auf keine der Beschreibungen
- Ähnlichkeit zu Block-Devices

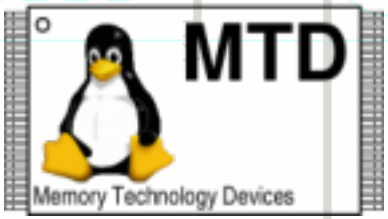
→ ***MTD***

MTD
(Memory Technology Devices)



	Block Device	MTD Device
Aufbau	Besteht aus Sektoren	Besteht aus „eraseblocks“
Größe der Sektoren/„eraseblocks“	512 – 1024 Bytes	Üblicherweise 128KB
Operationen	read sector write sector	read from eraseblock, write to eraseblock, erase eraseblock
Defekte Sektoren/„eraseblocks“	neu abgebildet und versteckt (Hardware)	werden nicht versteckt, sollten mit Software behandelt werden
Abnutzung	frei	NAND: 10^3 NOR: 10^5

MTD
(Memory Technology Device)



- **MTD User Modules**
  - Raw character access
  - Raw block access
  - Flash Translation Layer (FTL)
  - NAND Flash Translation Layer (NFTL)
  - Journaling Flash File System v2 (JFFS2)
- **MTD Hardware Device Drivers**
  - On-board memory
  - PCMCIA Devices
  - Onboard NOR-Flash (CFI)
  - Onboard NAND-Flash
  - M-Systems' DiskOnChip 2000 and Millenium
- **http://www.linux-mtd.infradead.org**

RAMFS / TMPFS

- „Partition“ im Arbeitsspeicher anlegen
  → Dateioperationen schneller (RAM)
- **Mount:**

```
mount -t tmpfs -o size=20m tmpfs /mnt/tmp

mount -t ramfs -o size=20m ramfs /mnt/ram
```

- **Unterschiede RAMFS/TMPFS:**

Experiment	RAMFS	TMPFS
Überschreiben	Ja	Nein → Error
Fixe Größe	Nein	Ja
Benutzt SWAP	Nein	Ja
Flüchtig	Ja	Ja

RAMFS / TMPFS

- **Nachteile von RAMFS und TMPFS**
  - Die Daten sind flüchtig
    → *Bei Shutdown/Breakdown sind Daten verloren*

    WORKAROUND:

    Routine die in bestimmten Intervallen oder beim Shutdown die Daten auf Platte sichert.

- **TMPFS unter Linux**

```
// Ausführung von init-Scripts
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)

// shared memory
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

http://wiki.debian.org/ramfs
http://en.wikipedia.org/wiki/TMPFS

CRAMFS

- CRAMFS (Compressed ROM Filesystem) ist ein freies unter der GPL stehendes *Read-Only-Dateisystem* mit integrierter Datenkompression
- Hauptsächlicher Einsatz bei „*embedded systems*“, darum ist der Hauptaugenmerk auf Einfachheit und Effizienz des benötigten Speicherplatzes gelegt
- Dateien sind mit der *zlib* komprimiert. Die Metainformationen sind unkomprimiert, werden jedoch in knapperer Struktur repräsentiert als bei konventionellen Dateisystemen.

CRAMFS

- *mkcramfs* zum Erstellen des Dateisystems und Aufnehmen von Dateien

  http://sourceforge.net/projects/cramfs/

- Bearbeitung unter Windows: *NewTuxFlashTools*

  http://www.dbox2.info/files/cat5/newtuxflashtools.zip

- Einschränkungen
  - Dateigröße max. 16MB
  - max. Systemgröße 256MB
  - nur unteren 8Bit der GID
  - Ordner “.“ & “..“ fehlen
  - keine Zeitstempel → 1970 GMT
  - gleicher Endian bei read/write

  - Kernel muss PAGE_CACHE_SIZE = 4096 besitzen

WORKAROUND (mkcramfs.c):

```
/*The kernel assumes PAGE_CACHE_SIZE as block size.*/
#define PAGE_CACHE_SIZE (4096)
```

CRAMFS / SQUASHFS

- SquashFS kann CRAMFS ersetzen und bietet zudem noch Vorteile:
  - speichert komplett UID/GID und Zeit der Dateierstellung
  - Dateien bis 16 Exabyte (10^{18}) [1.000.000 Terabyte]
  - redundante Dateien werden nur einmal gespeichert
  - Big- und Little-Endian Unterstützung

  http://squashfs.sourceforge.net/

- SquashFS wird häufig mit UnionFS verwendet um temporär Schreibzugriff auf Dateien zu erhalten
  - Einsatzzweck: Überlagerung von schreibgeschützten Dateisystemen.

  http://aufs.sourceforge.net/

JFFS2

(Journalling Flash File System version 2)

- 2001 wurde aufgrund hoher Nachfrage JFFS komplett neu geschrieben → JFFS2
- JFFS2 ist seit Kernelversion 2.4.10 eingebunden. Befindet sich auch im *Uboot* und *RedBoot* Bootloader.
- **Features:**
  - NAND-Flash Unterstützung
  - Hard Links
  - Kompression (zlib, rubin und rtime)
  - Bessere Performance

http://sources.redhat.com/jffs2/

JFFS2
(Journalling Flash File System version 2)

- **Nachteile:**
  - Es müssen alle „*nodes*“ beim einhängen gescant werden
  - Beim Schreiben vieler kleiner Datenblöcke kann es zu einer negativen Kompression kommen
  - Es gibt keine Möglichkeit zu sagen wieviel freier Speicher noch zur Verfügung steht
  - Es ist nicht bewiesen dass die Garbage Collection „*Dead Lock*“-frei arbeitet
  - Problem der sogenannten „*read & write*“-disturbs

LOGFS

- LOGFS = *log-strukturiertes skalierbares Flash Dateisystem*
- LOGFS zielt darauf ab JFFS2 in den meisten Fällen abzulösen
- Der Fokus liegt jedoch auf Geräten mit hohen Kapazitäten (> 64MB – 512MB)
- Entwickler: Jörn Engel (*http://logfs.org/logfs/*)
- Supporter: CELF (Consumer Electronics Linux Forum) *http://www.celinuxforum.org/*
- Im Gegensatz zu JFFS2, UBIFS und YAFFS bietet LOGFS die Möglichkeit, Block-Devices zu benutzen wie Solid-State-Drives, USB Flash Drives und Memory Cards

UBIFS

- UBIFS = *Unsorted Block Image File System*
- Entwickler: Universität von Szeged (Ungarn), Nokia
- Entwicklung seit 2007
- Seit Oktober 2008 erstmals im Linux Kernel 2.6.27
- UBIFS arbeitet auf der Basis von UBI-Inhalten, kann daher nicht auf der Basis von MTD verwendet werden

http://osl.sed.hu/wiki/ubifs

UBIFS

- Nachfolger von JFFS2
- Konkurrent von LOGFS

	JFFS2	UBIFS
Indexing auf Flash	Nein	B+ Baum mit „Wandering“-Algo.
Indexing im Speicher	Verkettete Liste, immer im Speicher	TNC (Tree Node Cache)
Schreiboperation	Schreibt Daten sofort *(write through)*	Cached die Daten beim Schreiben *(write back)*

YAFFS
(Yet Another Flash File System)

- YAFFS = *log-strukturiertes Dateisystem*
- Entwickelt nur für NAND-Flash
- YAFFS1 arbeitet mit alten Chips (512Byte/Seite)
- YAFFS2 arbeitet mit neueren Chips (2048Byte/Seite)
- YAFFS2 basiert auf YAFFS1
- OS: Linux, WinCE, pSOS, eCos, ThreadX, …
- YAFFS/Direct: kein OS, embedded OS und Bootloader
- Lizenz: GPL und Aleph One

http://www.yaffs.net/

Fragen?

http://www.fh-augsburg.de/~budda/elinux