

Dateien & Dateisysteme

PD Reinhard Bündgen
buendgen@de.ibm.com

Dateien in Unix

- Unix: alles ist eine Datei
 - Datensätze auf Hintergrundspeicher
 - Geräte
 - Netzwerkanschluss (Socket)
 - Kommunikationskanäle (pipes, fifos)
- Operationen auf Dateien
 - open/close, read/write, mmap, select/poll, lseek, fctl, ...
- Organisation von Dateien auf Hintergrundspeichern
 - Hierarchische Organisation der Dateien
 - Verzeichnisse, Dateisystembaum
 - Namespace, Dateisystem (Achtung überladene Bezeichnung)
- Literatur: Stevens & Rago: *Advanced Programming in the UNIX Environment* (2nd Ed.). Addison-Wsley 2005

Dateitypen in Unix

- - normale Datei
- d Verzeichnis
- c Character Device
- b Blockgerät
- l symbolischer Link
- p FIFO (named pipe)
- pipe
- socket

Dateisystem (DS): Begriffe

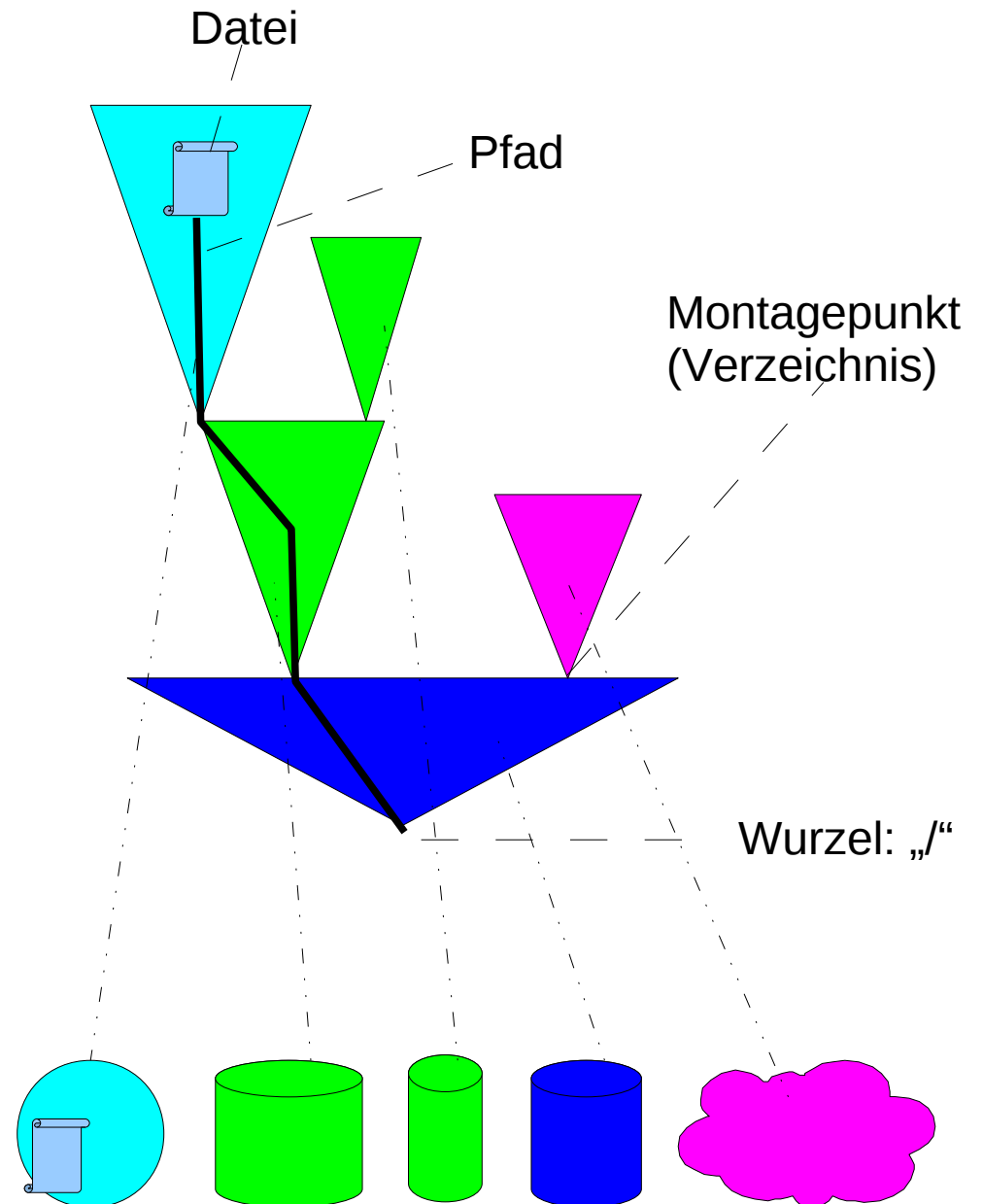
- Dateisystemtyp
 - Beschreibt wie Daten organisiert sind
- Dateisystem
 - Sammlung von Daten, die gemäß einem Dateisystemtyp organisiert sind
- Virtuelles DS (VFS)
 - Architekturelle Schicht im Kern die von spezifischen DS und DS-Typen abstrahiert
- Logisches DS, Dateisystembaum, Dateihierarchie, (namespace)
 - Benutzersicht aller adressierbaren Dateien
 - i.A. montiert aus verschiedenen Dateisystemen

Dateisystemtypen (Formate)

- Beschreibt
 - physische Organisation der Daten
 - Zugriffsweg zu Daten (via page cache?)
 - Implementierung der Dateisystem- und Dateioperationen
- Von Linux unterstützt (Auswahl)
 - HW DS:
 - minix, ext2/ext3/ext4, reiser, xfs, jfs, fat, hfs, hpfs, msdos, ntfs
 - formatieren mit `mkxxxfs` Kommando
 - Netzwerk DS
 - nfs, cifs, coda, afs, smbfs
 - Pseudo DS
 - proc, sys, debug
- Registrierung eines neuen Dateisystemtyps im Kern
 - `int register_filesystem(struct file_system_type * fs)`

Dateisystemmontage

- Mount Kommando
- mount() Systemruf
 - assoziiert formatierte Partition, Pseudo-DS oder NW DS mit Gebiet im logischen DS
 - DS Typ durch Option beschrieben (oft automatisch erkannt)
 - Optionen können weitere Eigenschaften beschreiben
 - Zugriffsautorisierung
 - Synchronisationsverhalten
 - ...
 - /etc/fstab
- (Montage) Namespace
 - Prozessspezifische Dateihierarchie

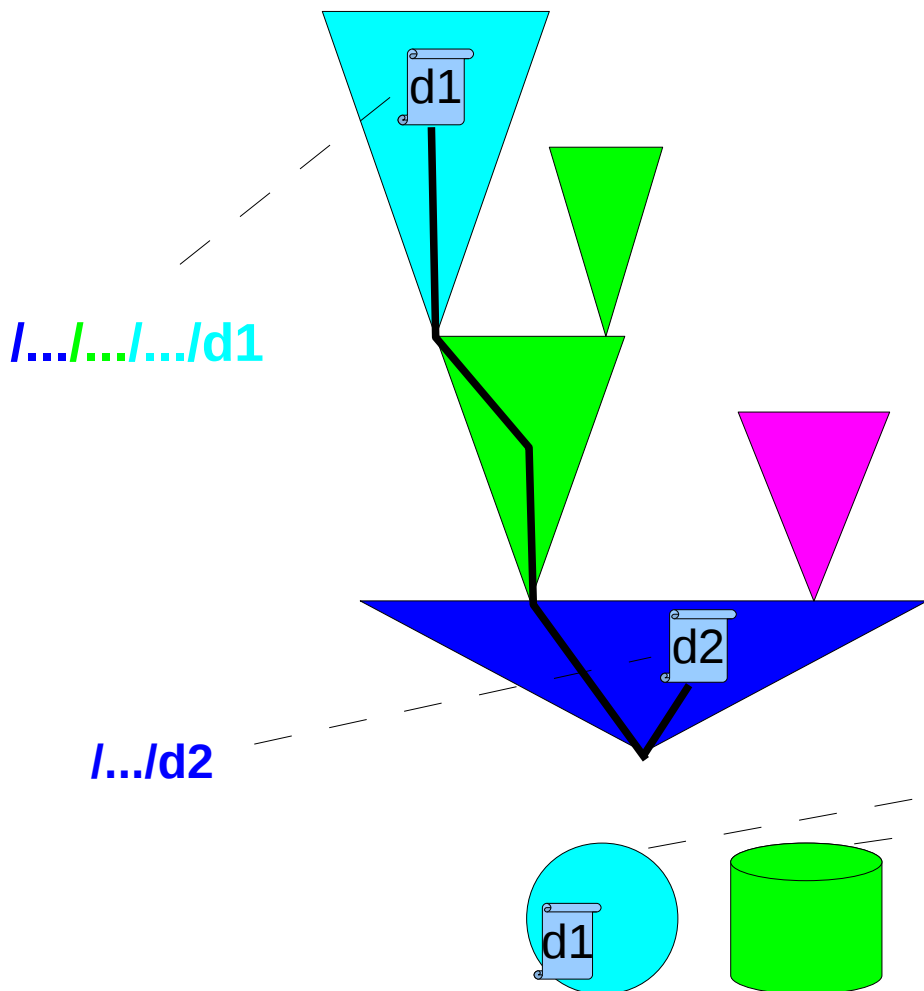


Das Virtuelle File System (VFS)

- Abstraktionsschicht im Kern
 - vermittelt zwischen Nutzerraum und konkreter Dateisystem Implementierung
- Gibt dem Nutzerraum eine einheitliche Schnittstelle auf verschiedenen Dateisystemtypen
 - stellt Standard Datei Systemrufe zur Verfügung
 - Implementierung der Funktion wird anhand der Dateisystemtypzugehörigkeit der zu manipulierenden Datei bestimmt
 - „objektorientiert“

VFS

Programme können
Dateioperationen
unabhängig vom
Speicherort und -typ
nutzen.



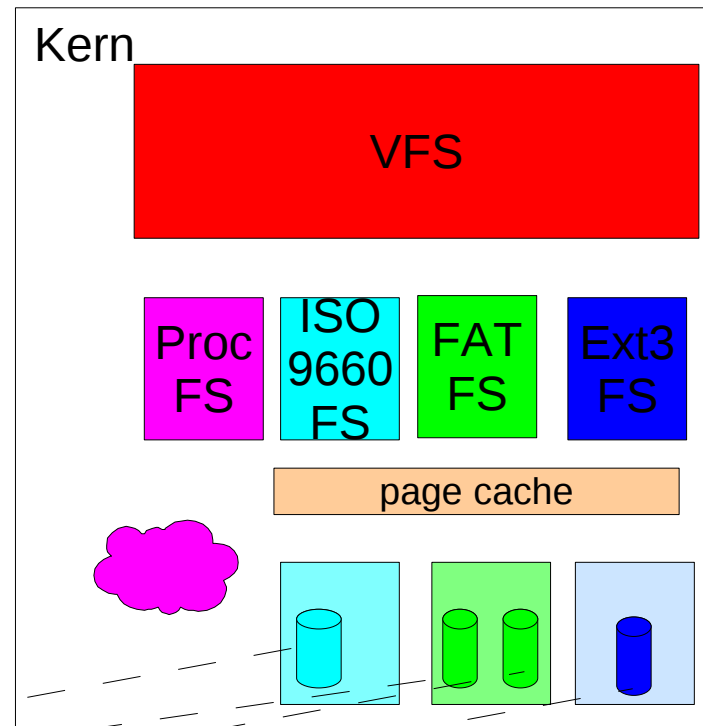
```
in=open(„/.../.../d1“,O_RDONLY,0);  
out=open(„/.../d2“,  
        O_WRONLY|O_CREATE|O_TRUNC,0600);  
do {  
    n = read(in,buf,4096);  
    write(out,buf,n);  
} while(n);  
close(out);  
close(in);
```

US: write()

VFS: sys_write()

FS: fs_write()

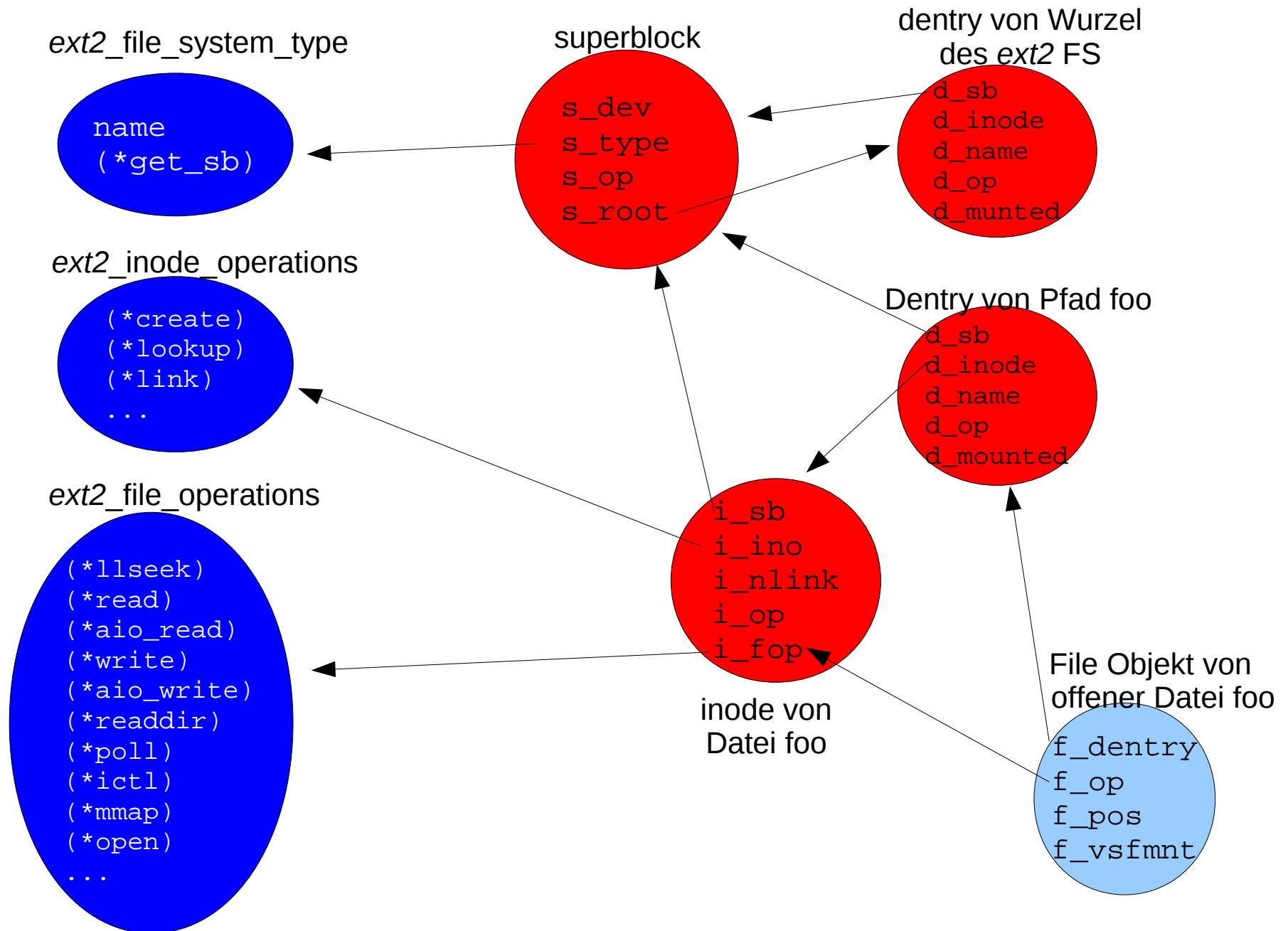
[DD: dd_write()]



Wichtige Objekte im VFS

- das VFS-Superblock-Objekt
 - `struct superblock (<linux/fs.h>)`
 - beschreibt montiertes FS, enthält u.a Info aus FS Control Block auf Platte
- das VFS-Inode-Objekt
 - `struct inode (<linux/fs.h>)`
 - identifiziert Datei im Kern, beschreibt Metadaten einer Datei, enthält u.a. Info aus File Control Block auf Platte (disk inode)
- das Dentry-Objekt
 - `struct dentry (<linux/dcache.h>)`
 - assoziiert Pfad mit Datei (Inode), gemäß Pfad ge-hash-t
- das Datei-Objekt (offene Datei)
 - `struct file (<linux/fs.h>)`
 - beschreibt Zustand einer offenen Datei einer Task

Wie hängen die Strukturen zusammen?



DS Montage im Kern

DS Montagen sind namespace spezifisch

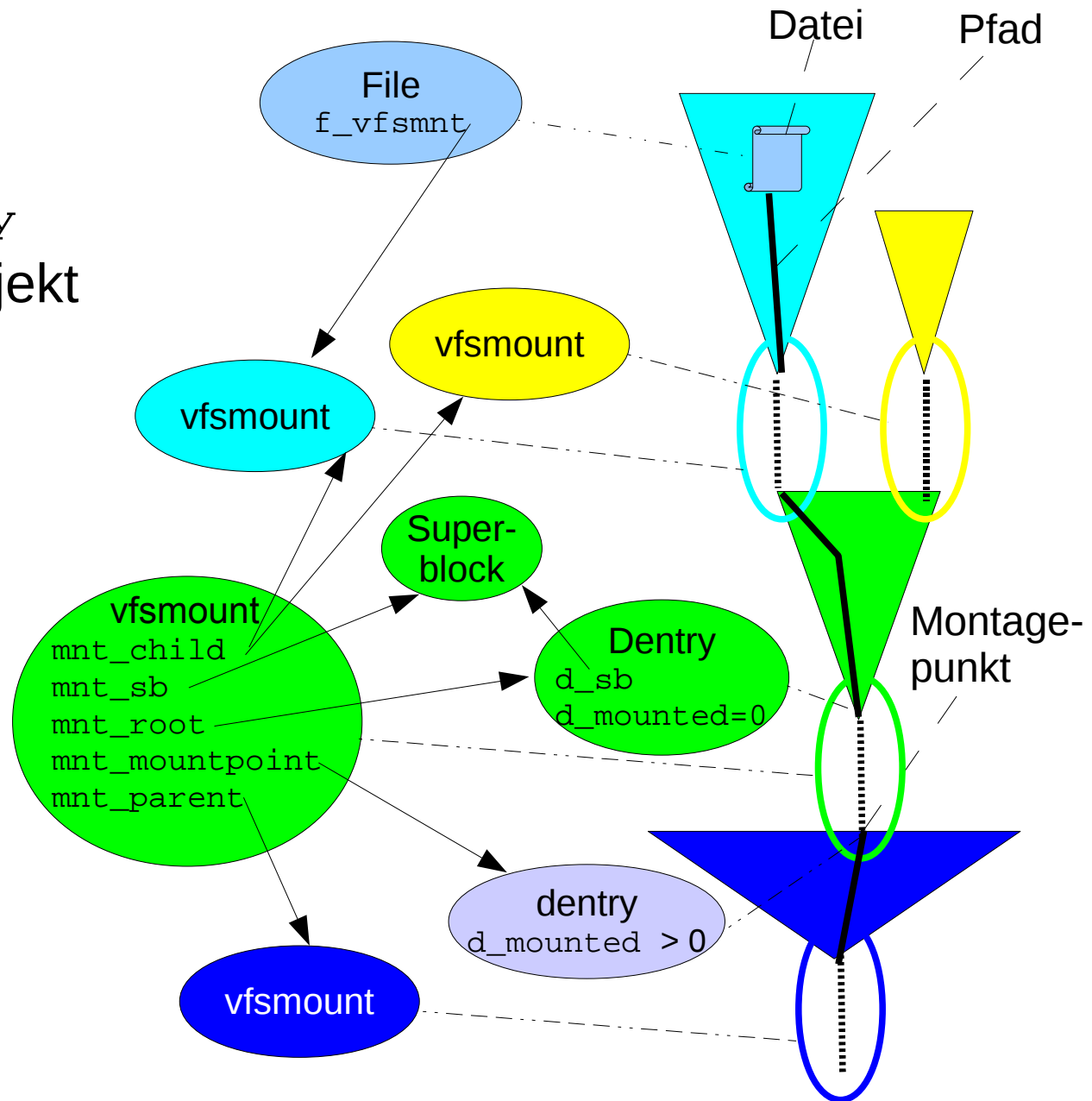
- `struct mnt_namespace`
- in `task_struct->nsproxy`

Beschrieben in Montageobjekt

- `struct vfsmount` (`<linux/mount.h>`):
 - Elterndateisystem
 - Kinderdateisysteme
 - Montagepunkt
 - Wurzel
 - Superblock
 - Namespace

Prozess spezifisch:

- `struct fs_struct` (`task_struct->fs`)
 - Enthält Montageobjekte für Wurzel und laufendes Verzeichnis



Offene Dateien in Unix

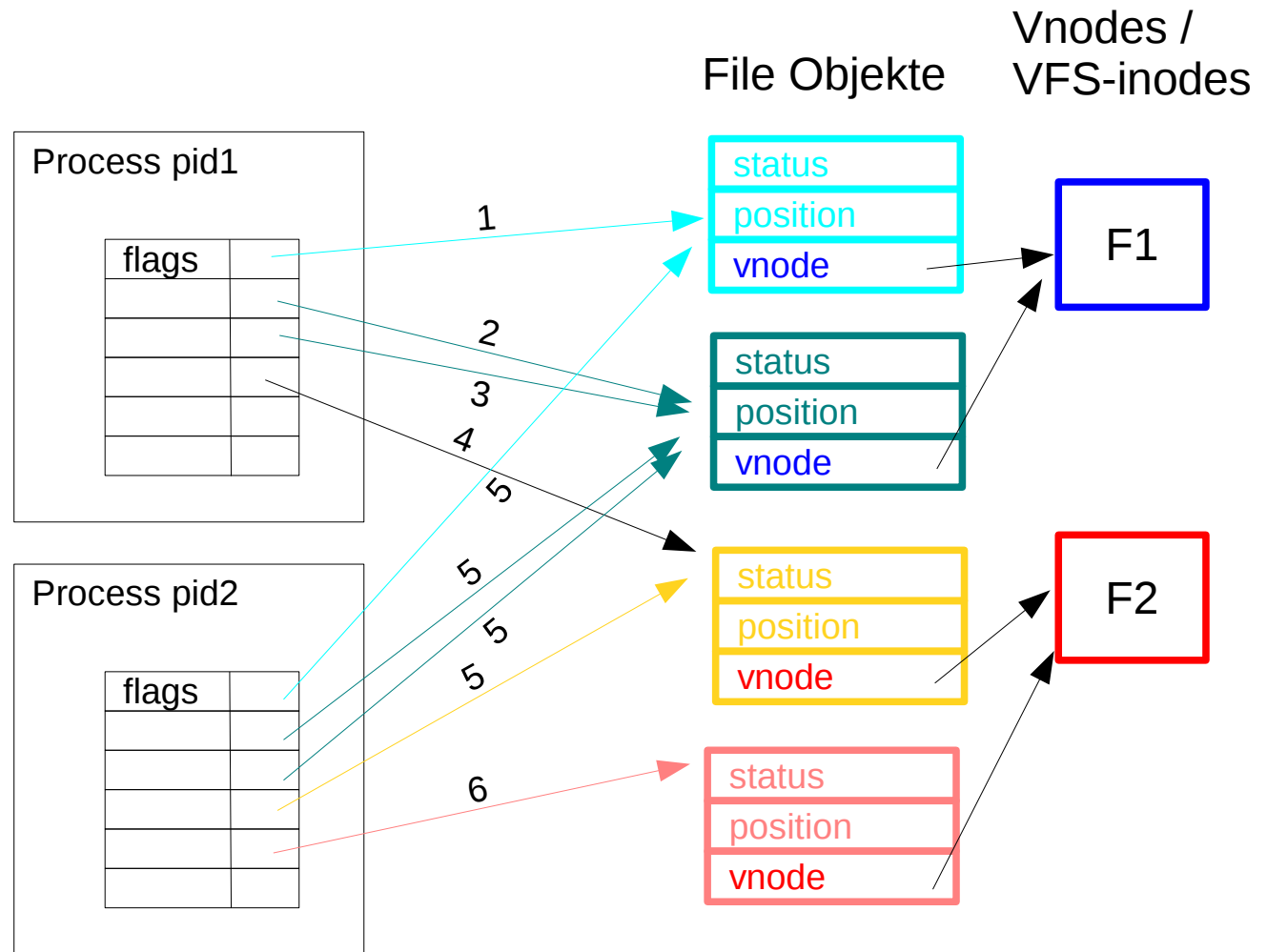
`fd=open(path, ...)`
assoziiert Pfad `path` mit
Dateibezeichner `fd`

`dup(fd)` erzeugt Kopie
eines Dateibezeichners

`fork()` kopiert alle
Dateibezeichner des
Vaterprozesses (file
sharing)

Dateibezeichner Flags:
`FD_CLOEXEC`

File Status: `open()`-Flags

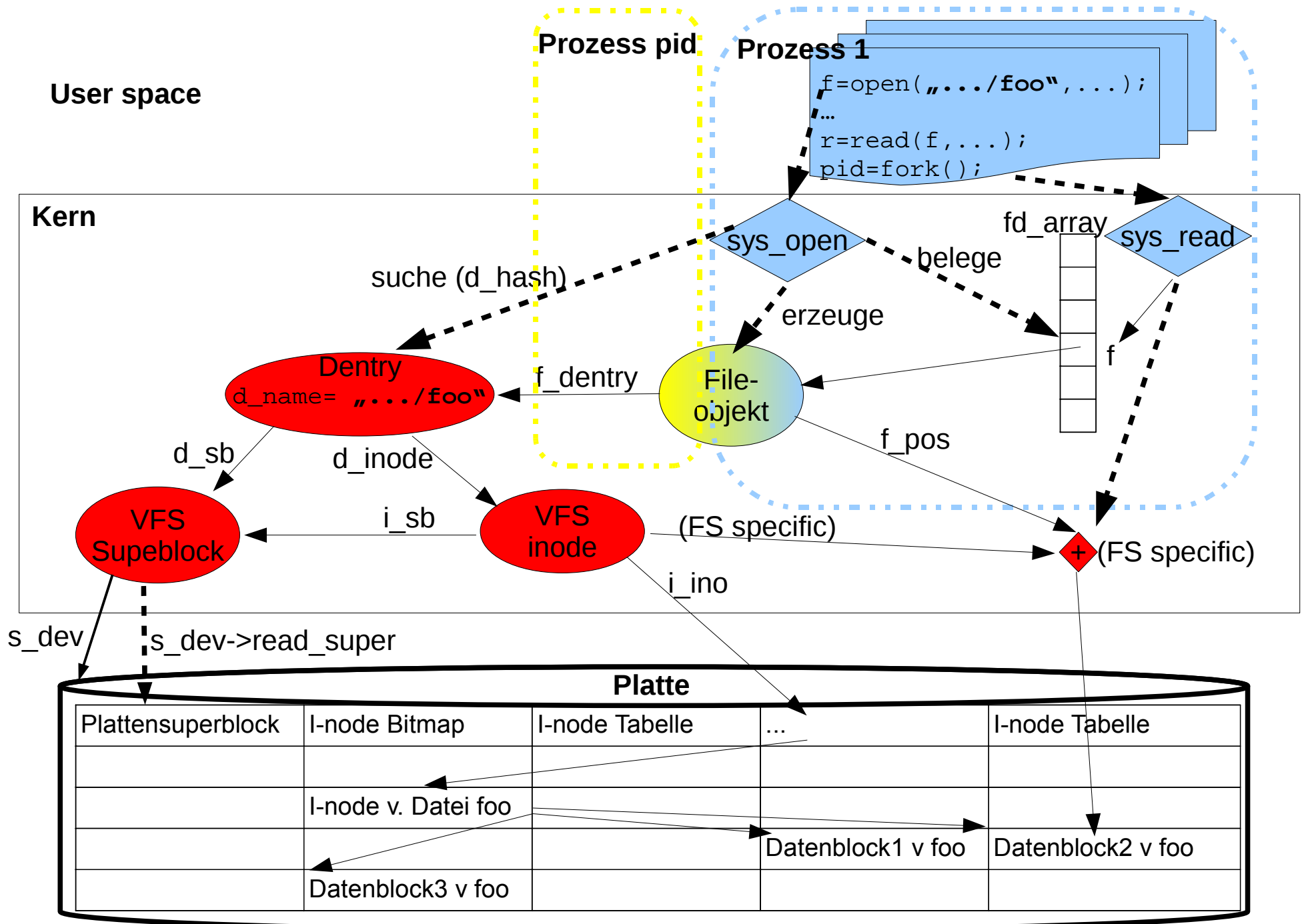


1. Process pid1: `fd0=open(F1,...);`
2. Process 1pid: `fd1=open(F1,...);`
3. Process pid1: `fd2=dup(fd1);`
4. Process pid1: `fd3=open(F2,...);`
5. Process pid1: `pid2=fork();`
6. Process pid2: `fd4=open(F2,...);`

Repräsentation von Dateien

- User:
 - Pfad, Dateiname, Dateityp
- Prozess (offene Datei)
 - Dateibezeichner (fd)
- Kerntasks (offene Datei)
 - Fileobjekt
- Kern
 - inode object, dentry
- Physisch (auf Partition)
 - Disk inode

Wie passt alles zusammen?



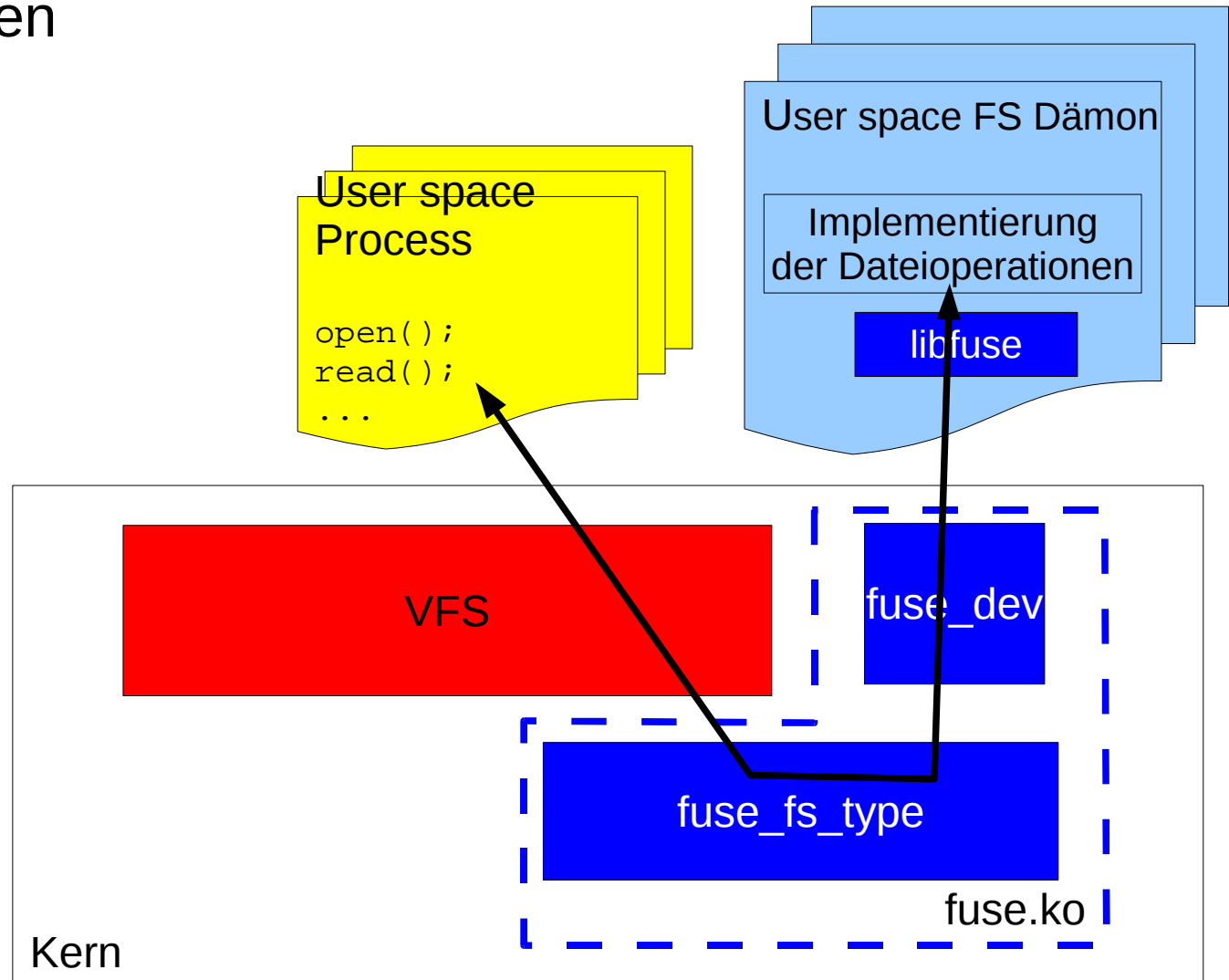
Filesystem in user space (FUSE)

FUSE Komponenten

- fuse.ko Modul
- libfuse
- fusermount

User Space FUSE Dämon

- Liest fuse Anfragen von fuse Gerät
 - /dev/fuse



Besondere Dateisysteme

- File System Check / Journaling
 - Metadaten / Daten
- Dynamische Veränderung der Größe
- Verteilte DS, Netzwerk DS
- Cluster DS
- User Space DS
- Crypto DS
- Device backed / Pseudo DS
- Nutzung von page cache oder nicht