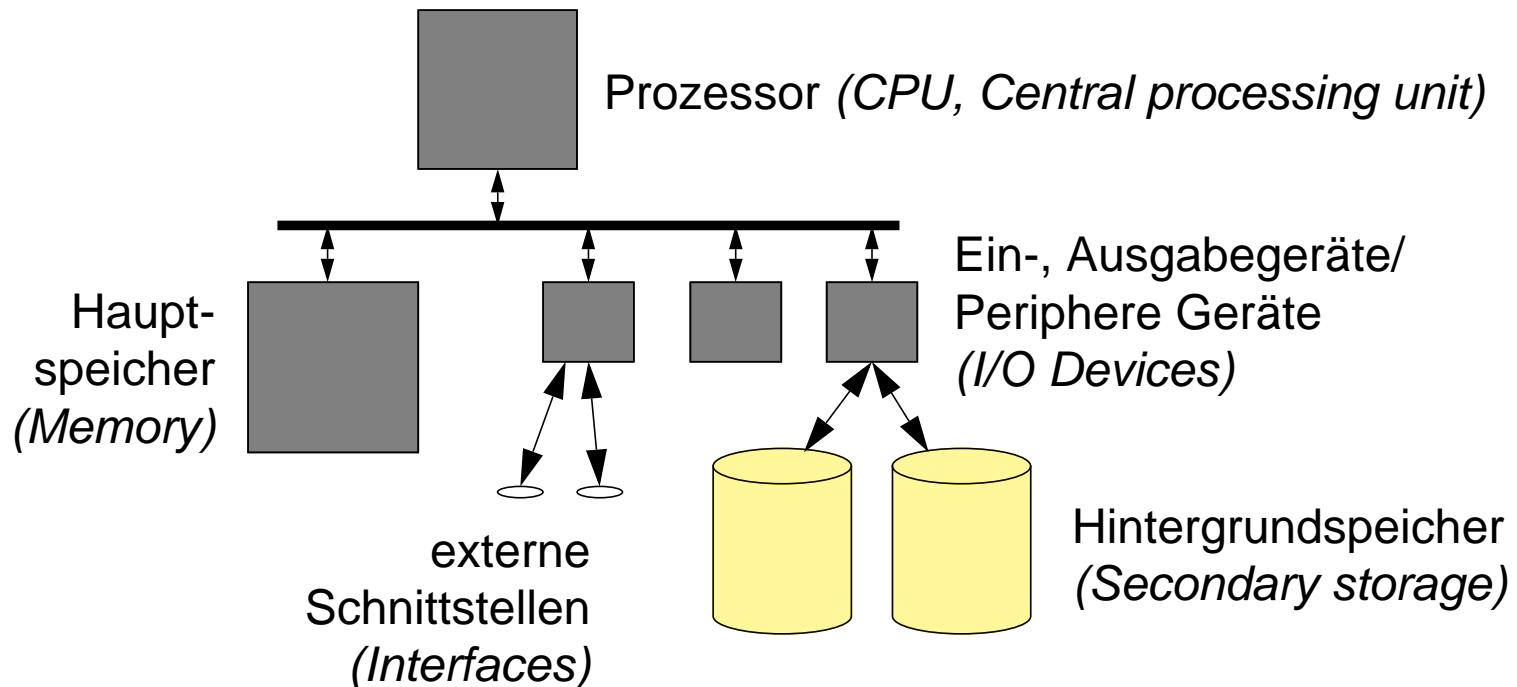


G Dateisysteme

G.1 Allgemeine Konzepte

■ Einordnung

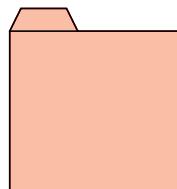


G.2 Allgemeine Konzepte (2)

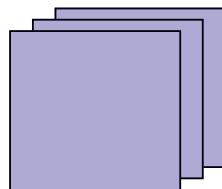
- Dateisysteme speichern Daten und Programme persistent in Dateien
 - ◆ Betriebssystemabstraktion zur Nutzung von Hintergrundspeichern (z.B. Platten, CD-ROM, Bandlaufwerke)
 - Benutzer muss sich nicht um die Ansteuerungen verschiedener Speichermedien kümmern
 - einheitliche Sicht auf den Hintergrundspeicher
- Dateisysteme bestehen aus
 - ◆ Dateien (*Files*)
 - ◆ Katalogen (*Directories*)
 - ◆ Partitionen (*Partitions*)

G.2 Allgemeine Konzepte (3)

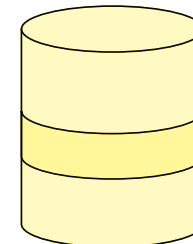
- Datei
 - ◆ speichert Daten oder Programme
- Katalog / Verzeichnis (*Directory*)
 - ◆ erlaubt Benennung der Dateien
 - ◆ enthält Zusatzinformationen zu Dateien
- Partitionen
 - ◆ eine Menge von Katalogen und deren Dateien
 - ◆ sie dienen zum physischen oder logischen Trennen von Dateimengen.



Katalog



Dateien



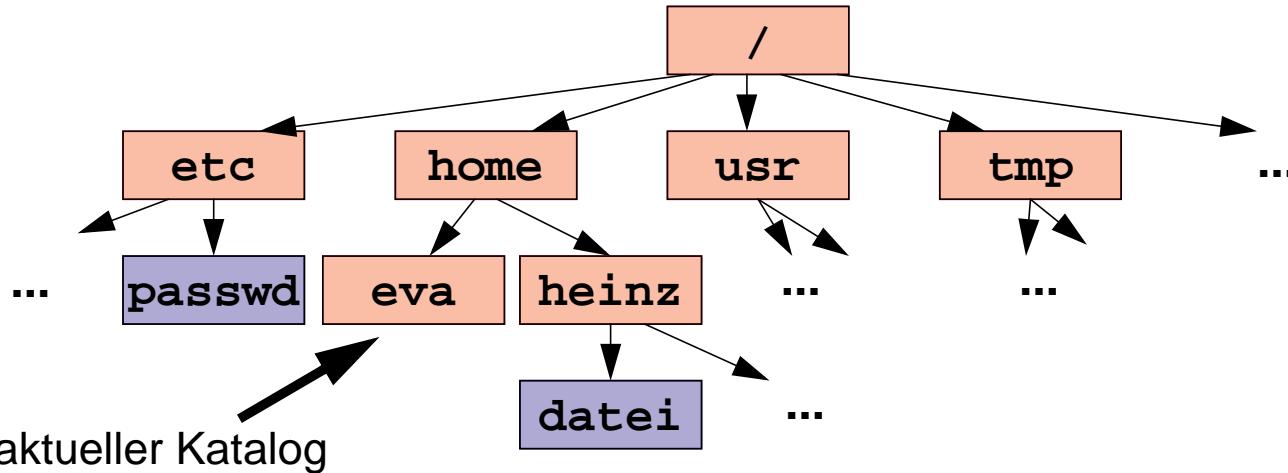
Partition

G.3 Beispiel: UNIX (Sun-UFS)

- Datei
 - ◆ einfache, unstrukturierte Folge von Bytes
 - ◆ beliebiger Inhalt; für das Betriebssystem ist der Inhalt transparent
 - ◆ dynamisch erweiterbar
- Katalog
 - ◆ baumförmig strukturiert
 - Knoten des Baums sind Kataloge
 - Blätter des Baums sind Verweise auf Dateien
 - ◆ jedem UNIX-Prozess ist zu jeder Zeit ein aktueller Katalog (*Current working directory*) zugeordnet
- Partitionen
 - jede Partition enthält einen eigenen Dateibaum
 - Bäume der Partitionen werden durch "mounten" zu einem homogenen Dateibaum zusammengebaut (Grenzen für Anwender nicht sichtbar!)

1 Pfadnamen

■ Baumstruktur

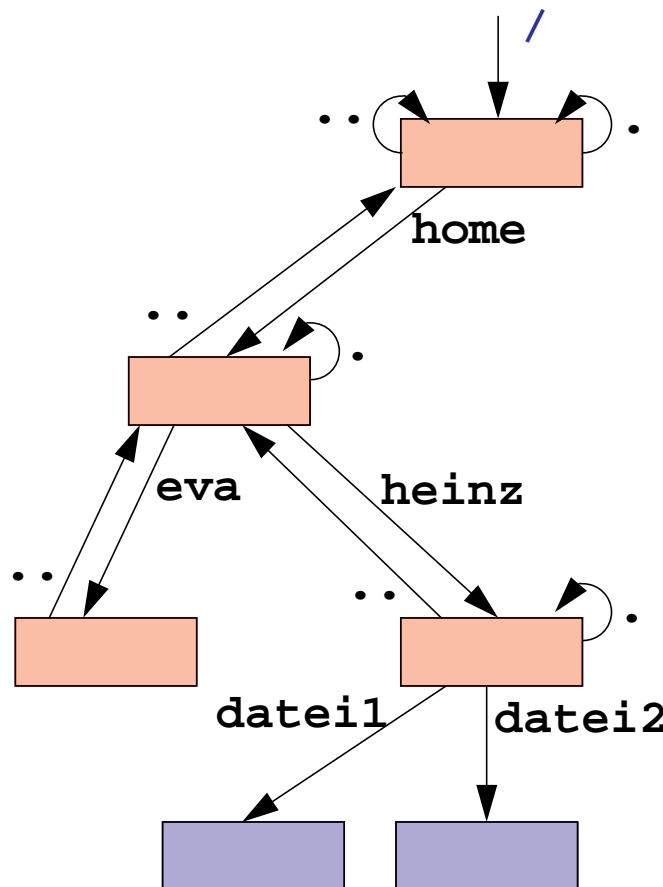


■ Pfade

- ◆ z.B. „/home/heinz/datei“, „/tmp“, „.../heinz/datei“
- ◆ „/“ ist Trennsymbol (*Slash*); beginnender „/“ bezeichnet Wurzelkatalog; sonst Beginn implizit mit dem aktuellem Katalog

1 Pfadnamen (2)

■ Eigentliche Baumstruktur



- ◆ benannt sind nicht Dateien und Kataloge, sondern die Verbindungen zwischen ihnen
 - ◆ Kataloge und Dateien können auf verschiedenen Pfaden erreichbar sein
z. B. `.../heinz/datei1` und `/home/heinz/datei1`
 - ◆ Jeder Katalog enthält
 - einen Verweis auf sich selbst (`.`) und
 - einen Verweis auf den darüberliegenden Katalog im Baum (`..`)
 - Verweise auf Dateien

2 Programmierschnittstelle für Kataloge

■ Kataloge verwalten

- ◆ Erzeugen

```
int mkdir( const char *path, mode_t mode );
```

- ◆ Löschen

```
int rmdir( const char *path );
```

■ Kataloge lesen (Schnittstelle der C-Bibliothek)

- Katalog öffnen:

```
DIR *opendir( const char *path );
```

- Katalogeinträge lesen:

```
struct dirent *readdir( DIR *dirp );
```

- Katalog schließen:

```
int closedir( DIR *dirp );
```

■ "eigentliche" Systemschnittstelle (open, getdents) wird normalerweise nicht direkt verwendet

2 Kataloge (2): opendir / closedir

■ Funktionsschnittstelle:

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir(const char *dirname);

int closedir(DIR *dirp);
```

■ Argument von opendir

- ◆ **dirname**: Verzeichnisname

■ Rückgabewert: Zeiger auf Datenstruktur vom Typ **DIR** oder **NULL**

2 Kataloge (3): readdir

- Funktionsschnittstelle:

```
#include <sys/types.h>
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
```

- Argumente

- ◆ **dirp**: Zeiger auf **DIR**-Datenstruktur

- Rückgabewert: Zeiger auf Datenstruktur vom Typ **struct dirent** oder **NULL** wenn fertig oder Fehler (**errno** vorher auf 0 setzen!)
- Probleme: Der Speicher für **struct dirent** wird von der Bibliothek wieder verwendet!

2 Kataloge (4): struct dirent

- Definition unter Linux (/usr/include/bits/dirent.h)

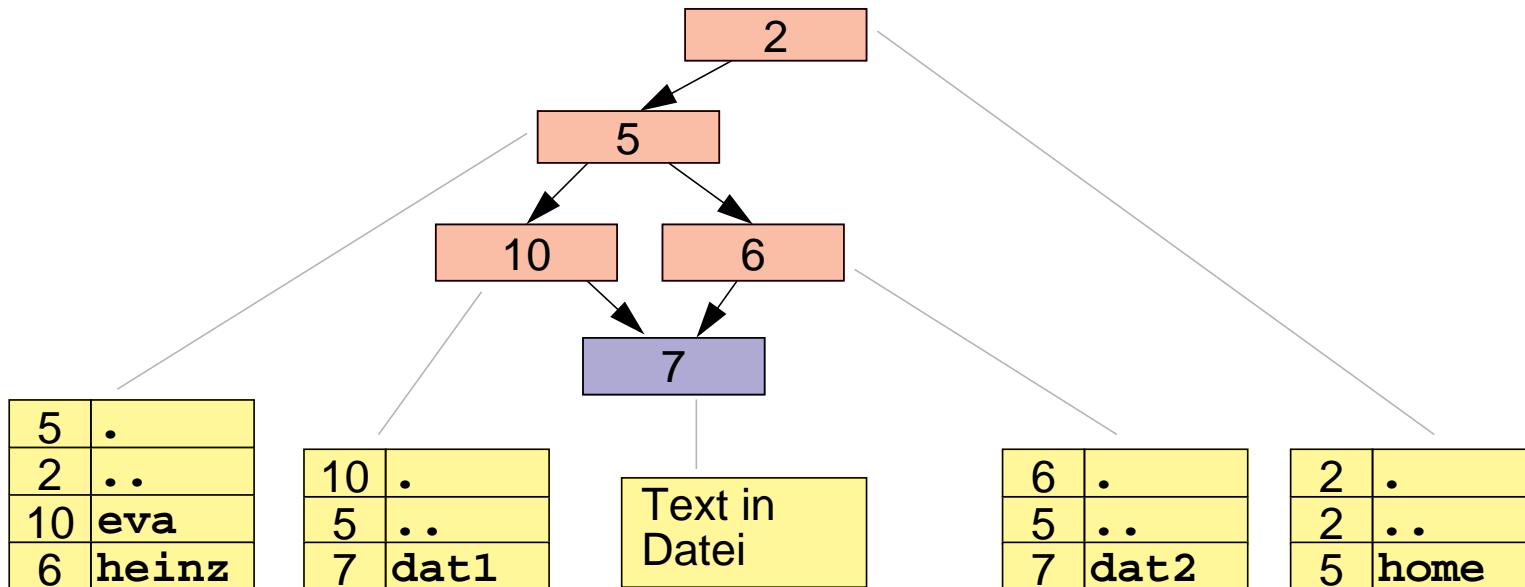
```
struct dirent {
    __ino_t d_ino;
    __off_t d_off;
    unsigned short int d_reclen;
    unsigned char d_type;
    char d_name[256];
};
```

3 Programmierschnittstelle für Dateien

- siehe C-Ein/Ausgabe (Schnittstelle der C-Bibliothek)
- C-Funktionen (fopen, printf, scanf, getchar, fputs, fclose, ...) verbergen die "eigentliche Systemschnittstelle und bieten mehr "Komfort"
 - open, close, read, write

4 Inodes

- Attribute (Zugriffsrechte, Eigentümer, etc.) einer Datei und Ortsinformation über ihren Inhalt werden in **Inodes** gehalten
 - ◆ Inodes werden pro Partition numenriert (*Inode number*)
- Kataloge enthalten lediglich Paare von Namen und Inode-Nummern
 - ◆ Kataloge bilden einen hierarchischen Namensraum über einem eigentlich flachen Namensraum (durchnumenrierte Dateien)



4 Inodes (2)

■ Inhalt eines Inode

- ◆ Dateityp: Katalog, normale Datei, Spezialdatei (z.B. Gerät)
- ◆ Eigentümer und Gruppe
- ◆ Zugriffsrechte
- ◆ Zugriffszeiten: letzte Änderung (*mtime*), letzter Zugriff (*atime*), letzte Änderung des Inodes (*ctime*)
- ◆ Anzahl der Hard links auf den Inode
- ◆ Dateigröße (in Bytes)
- ◆ Adressen der Datenblöcke des Datei- oder Kataloginhalts