Principper for Samtidighed og Styresystemer

Filer og filsystemer

René Rydhof Hansen

Februar 2008

Mål

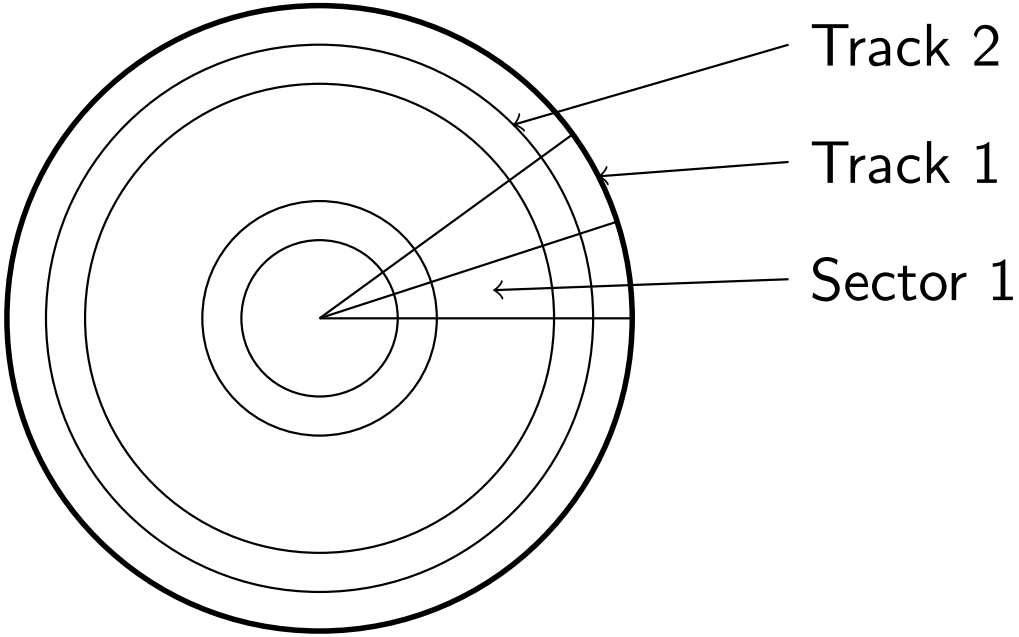
At kunne

- ... forklare hvad og hvordan et filsystem abstraherer over
- ... forklare grundlæggende organisationsprincipper for filsystemer
- ... forklare hard-links vs. symboli-links
- ... forklare hvad et virtuelt filsystem er
- ... forklare grundlæggende egenskaber og operationer ved filsystemer (filtyper, låse, rettigheder)

Opgaver

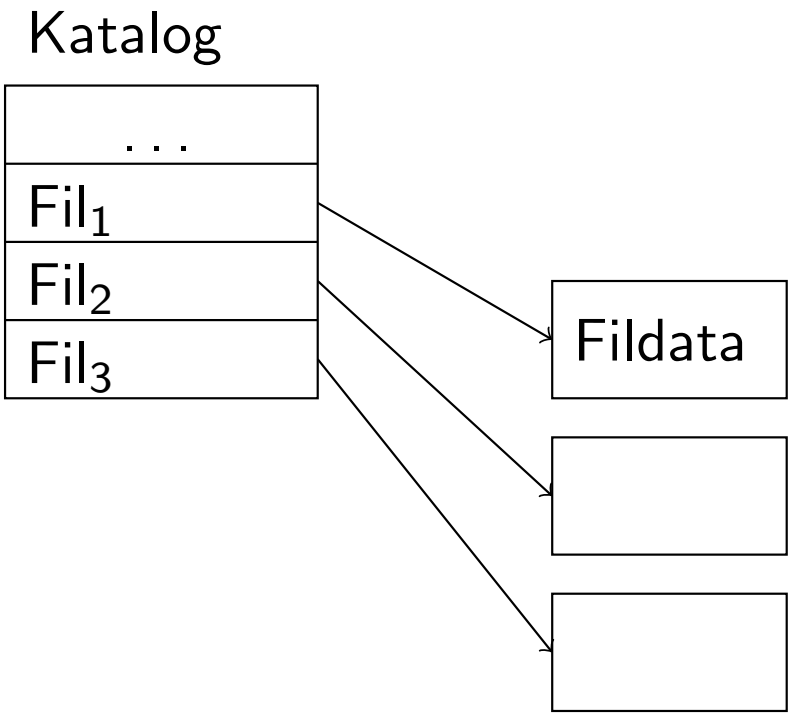
- Opgave 1: Grundlæggende begreber
- Opgave 2: Fordele/ulemper ved monolitisk hhv. mini-kerne design
- Opgave 3: Turing-ækvivalens
- Opgave 4: TOCTTOU

Filsystemer



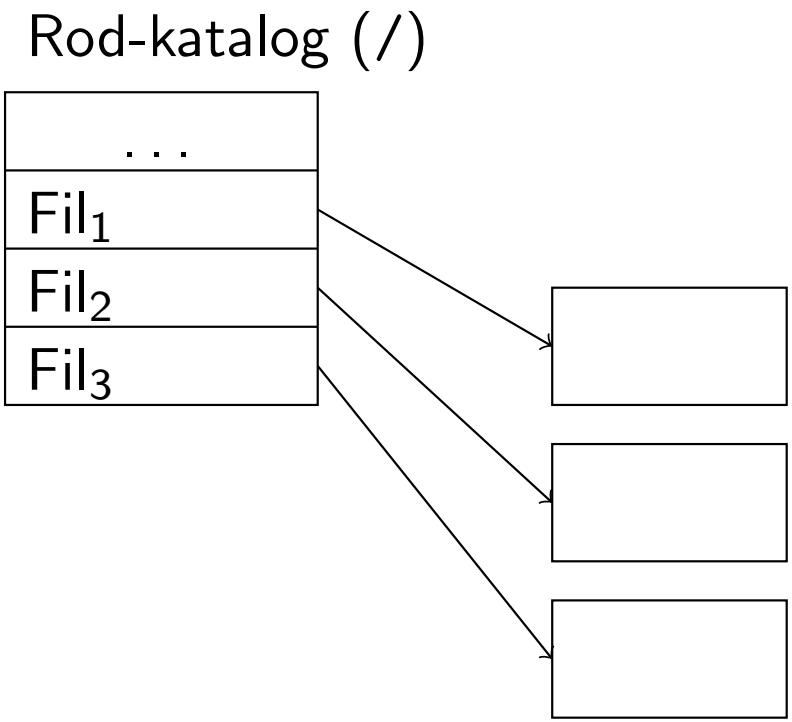
- Oprettelse og sletning af filer
- Navngivning
- Læsning og skrivning
- Deling og beskyttelse
- Pålidelighed og fejltolerance
- Effektivitet
- Pladsudnyttelse

Navnerum



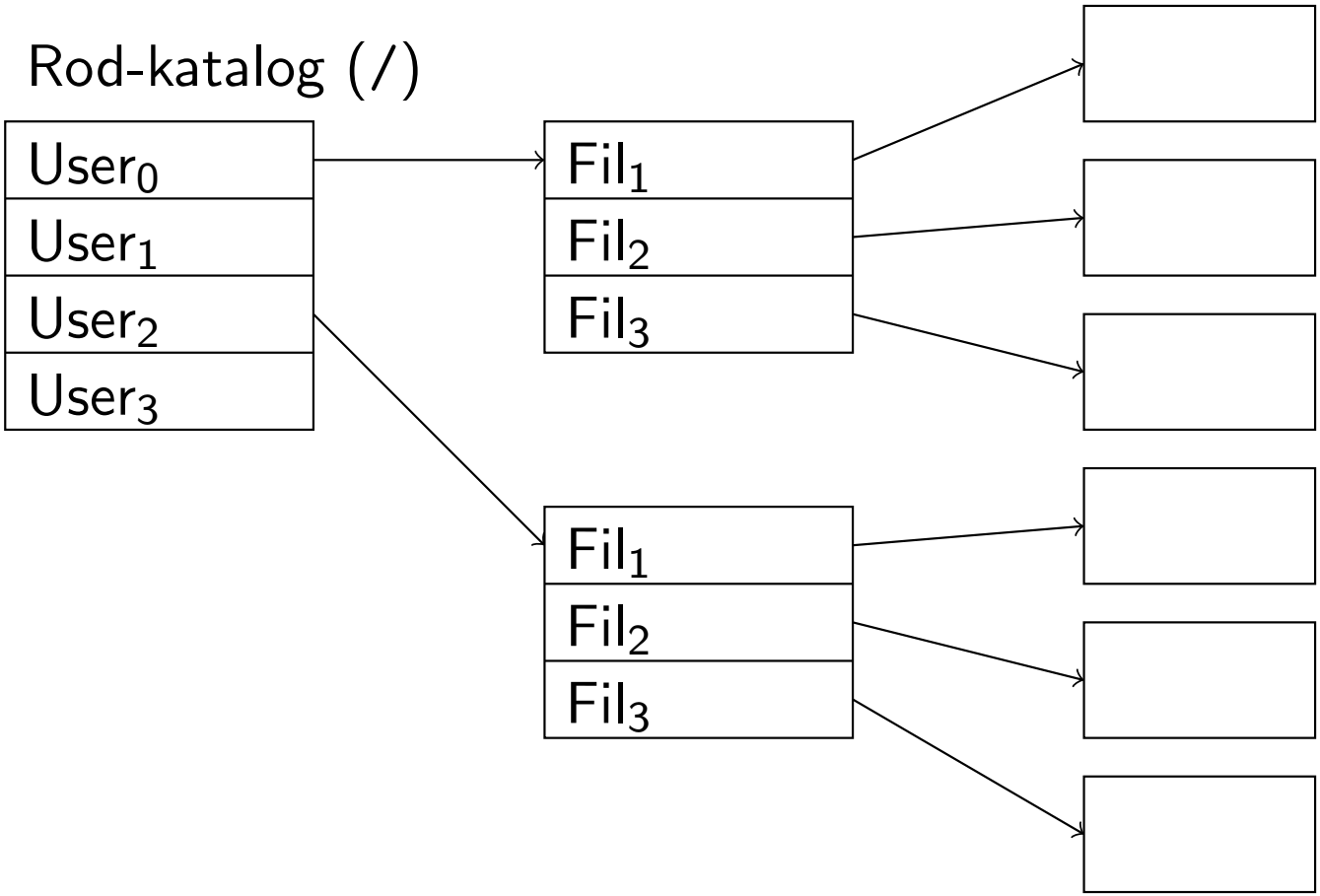
- Filsystemer definerer navnerum for filer og kataloger etc.
- Abstraktion over fysisk placering

Flade filsystemer



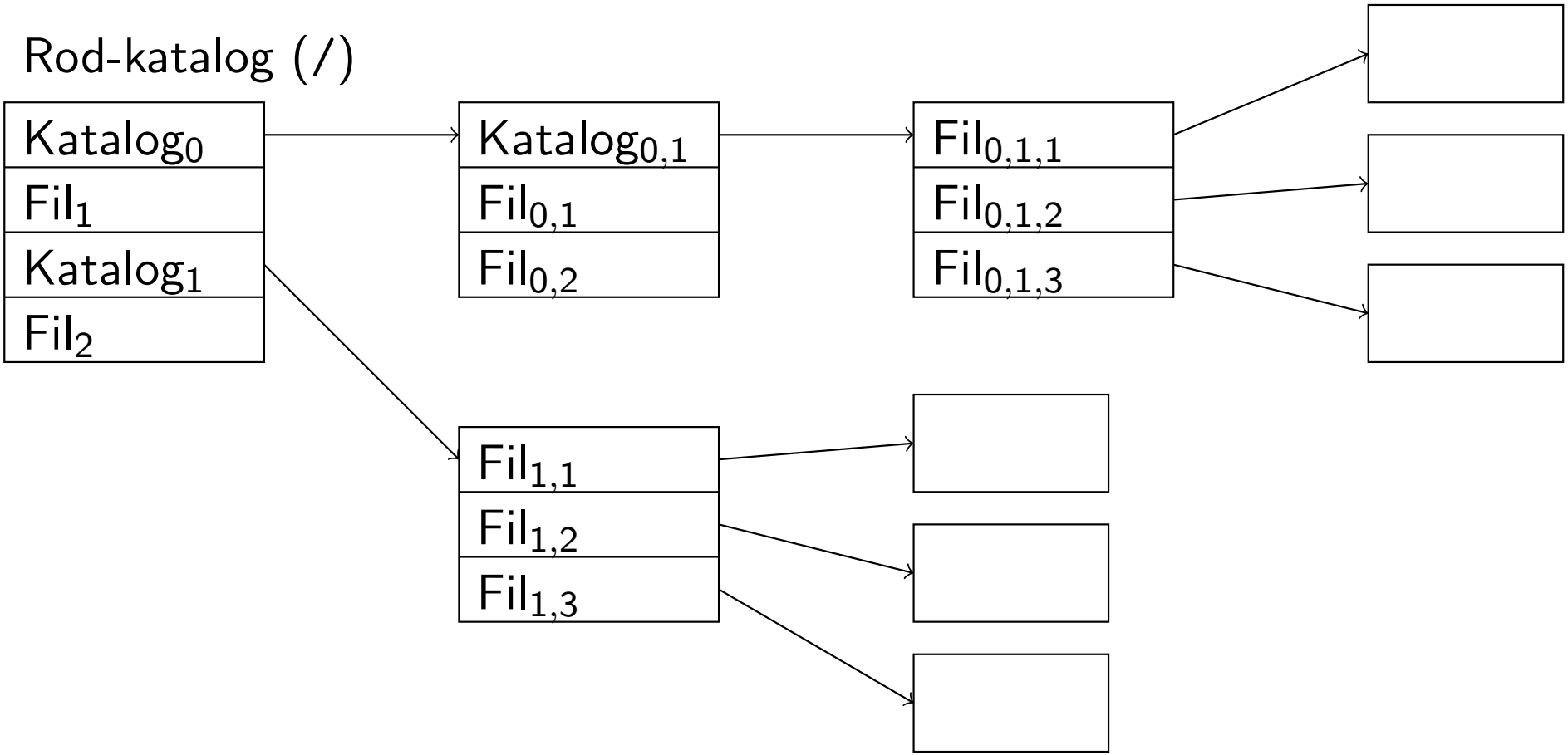
- Kun eet katalog
- Tidlig MS-DOS(?)

To-niveau filsystemer



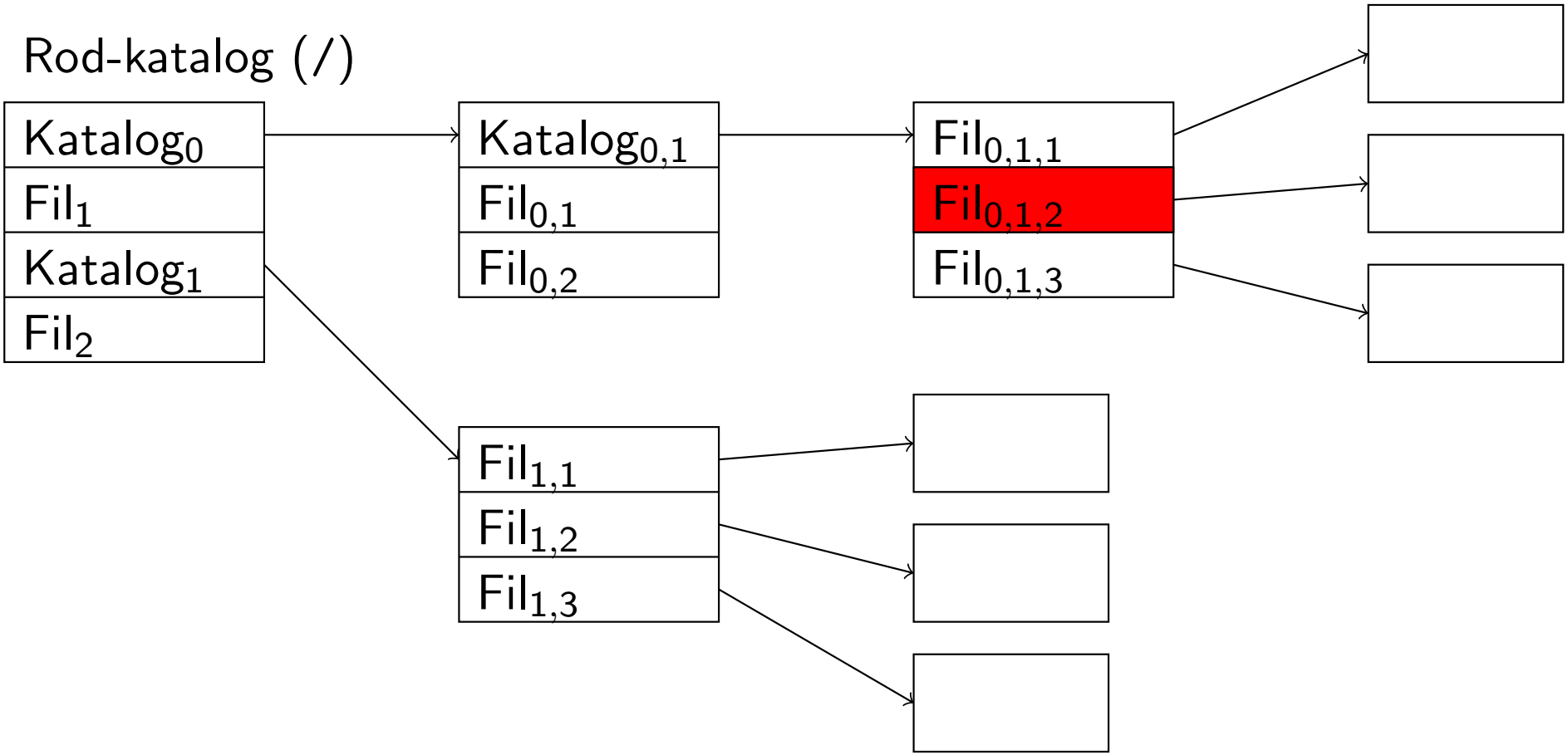
- F.eks. et katalog per bruger
- CP/M (16 user areas)

Hierarkiske filsystemer



- Organisering i kataloger
- Kataloger organiseret som: træer, acykliske orienterede grafer, orienterede grafer

Hierarkiske filsystemer



- Organisering i kataloger
- Kataloger organiseret som: træer, acykliske orienterede grafer, orienterede grafer

Stier

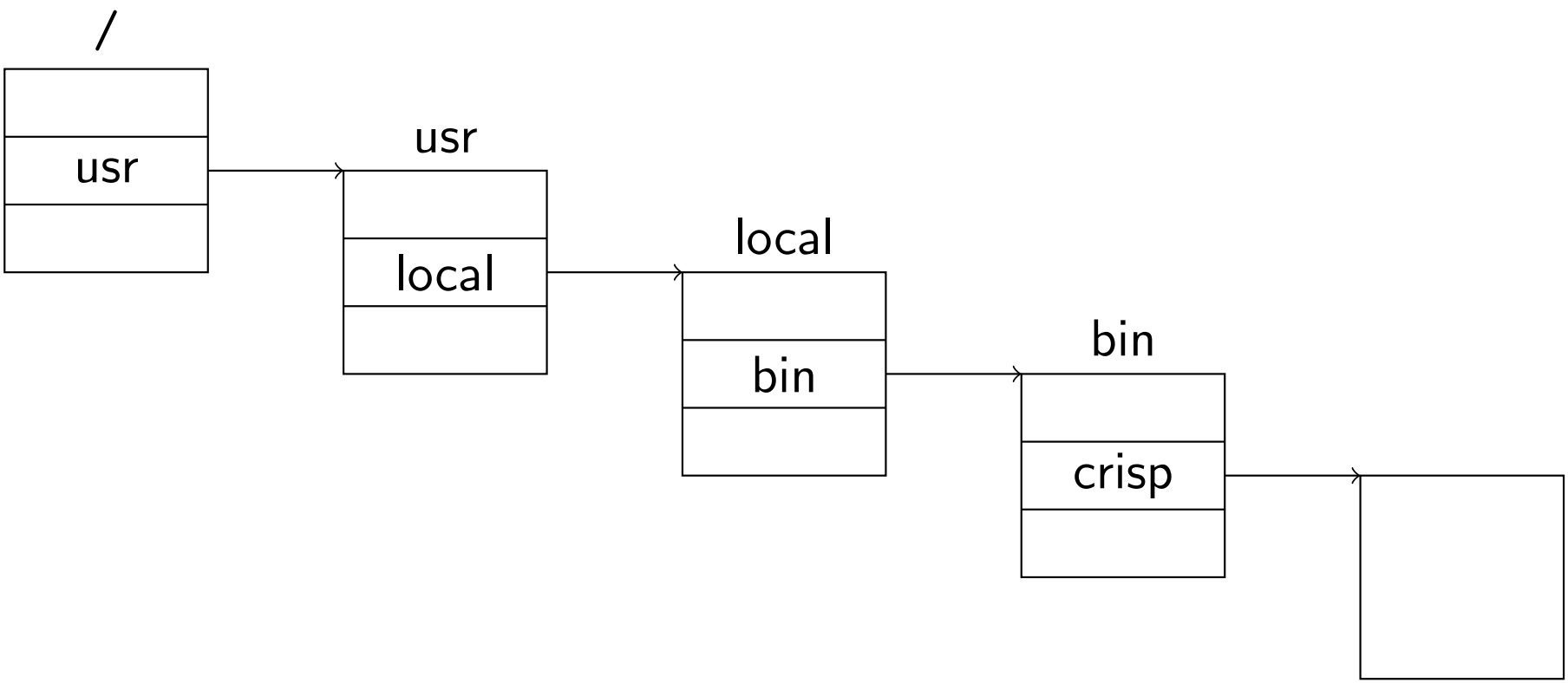
- Navngivning: $\text{Fil}_{0,1,2}$
  - Sti: $/\text{Katalog}_0/\text{Katalog}_1/\text{Fil}_2$
- Hvor mange kataloger er der i flg. sti?

`/usr/local/bin/crisp`

Stier

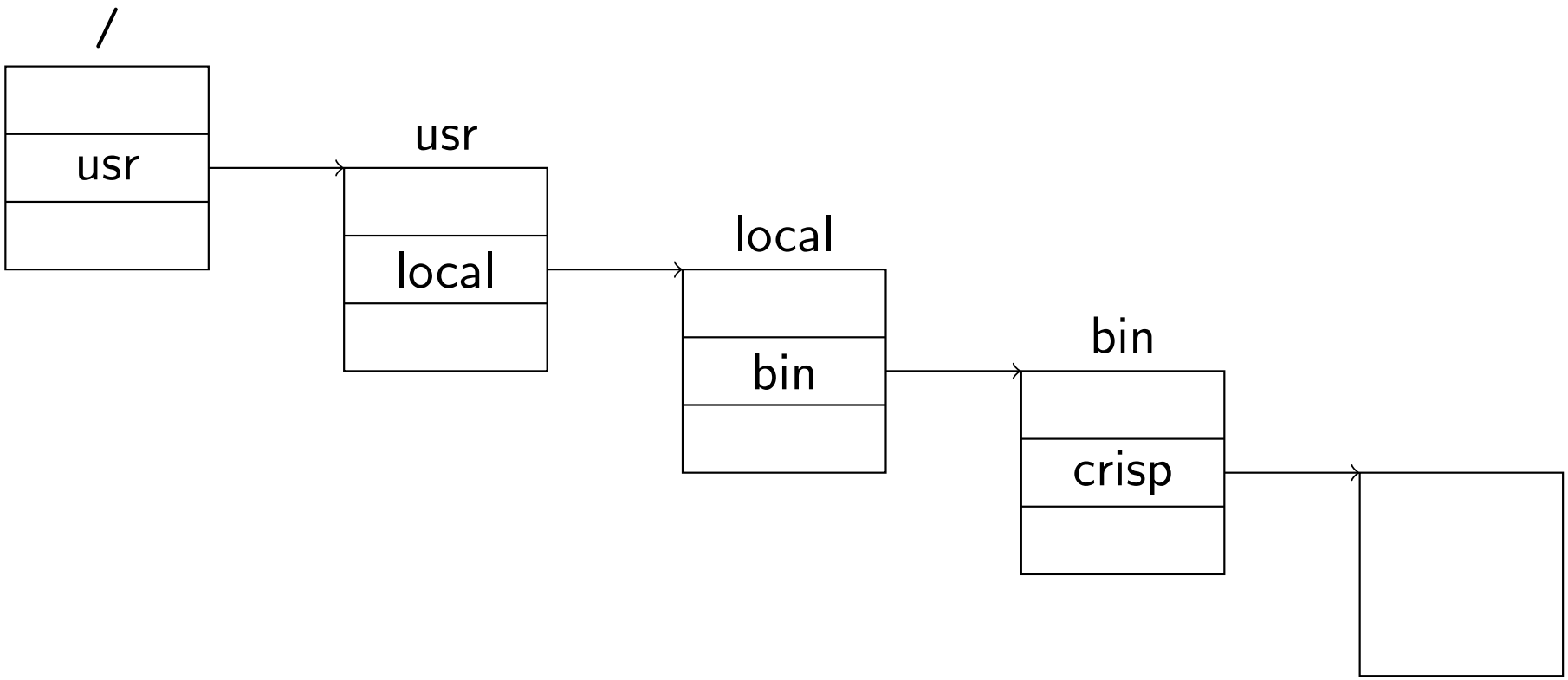
- Navngivning: $\text{Fil}_{0,1,2}$
  - Sti: $/\text{Katalog}_0/\text{Katalog}_1/\text{Fil}_2$
- Hvor mange kataloger er der i flg. sti?

`/usr/local/bin/crisp`



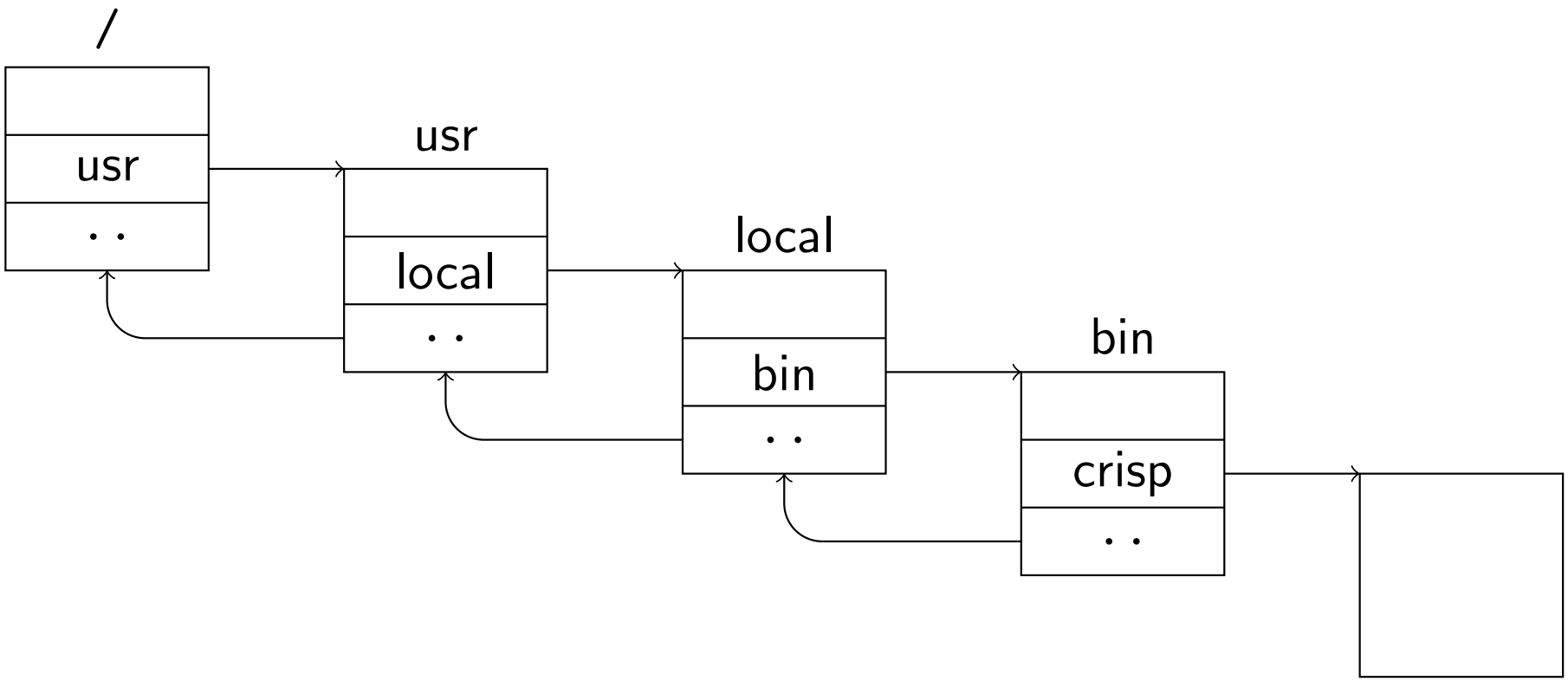
Cykler

- Hvordan kan cykler (i katalog-grafen) opstå?



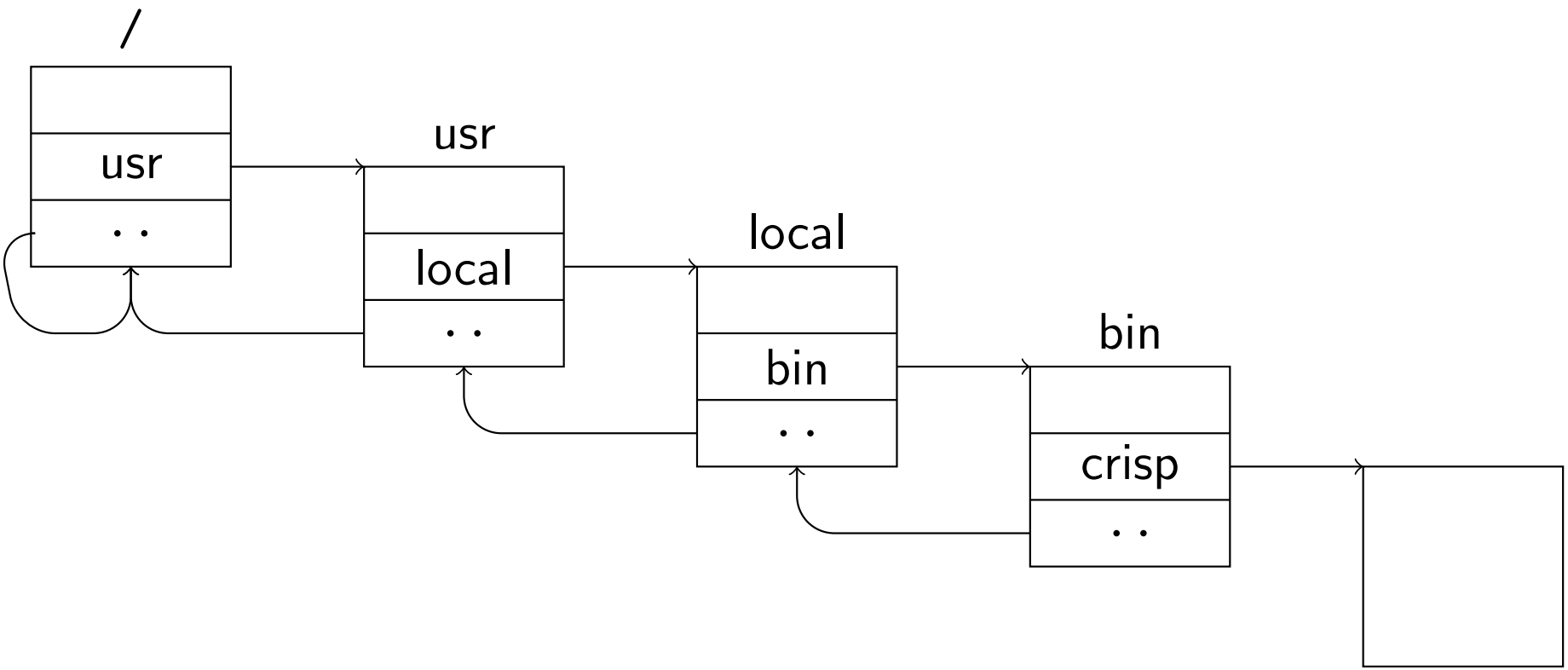
Cykler

- Hvordan kan cykler (i katalog-grafen) opstå?
  - '. .' peger på "forældre-kataloget"
  - Hvad med . . i rodkataloget?
  -



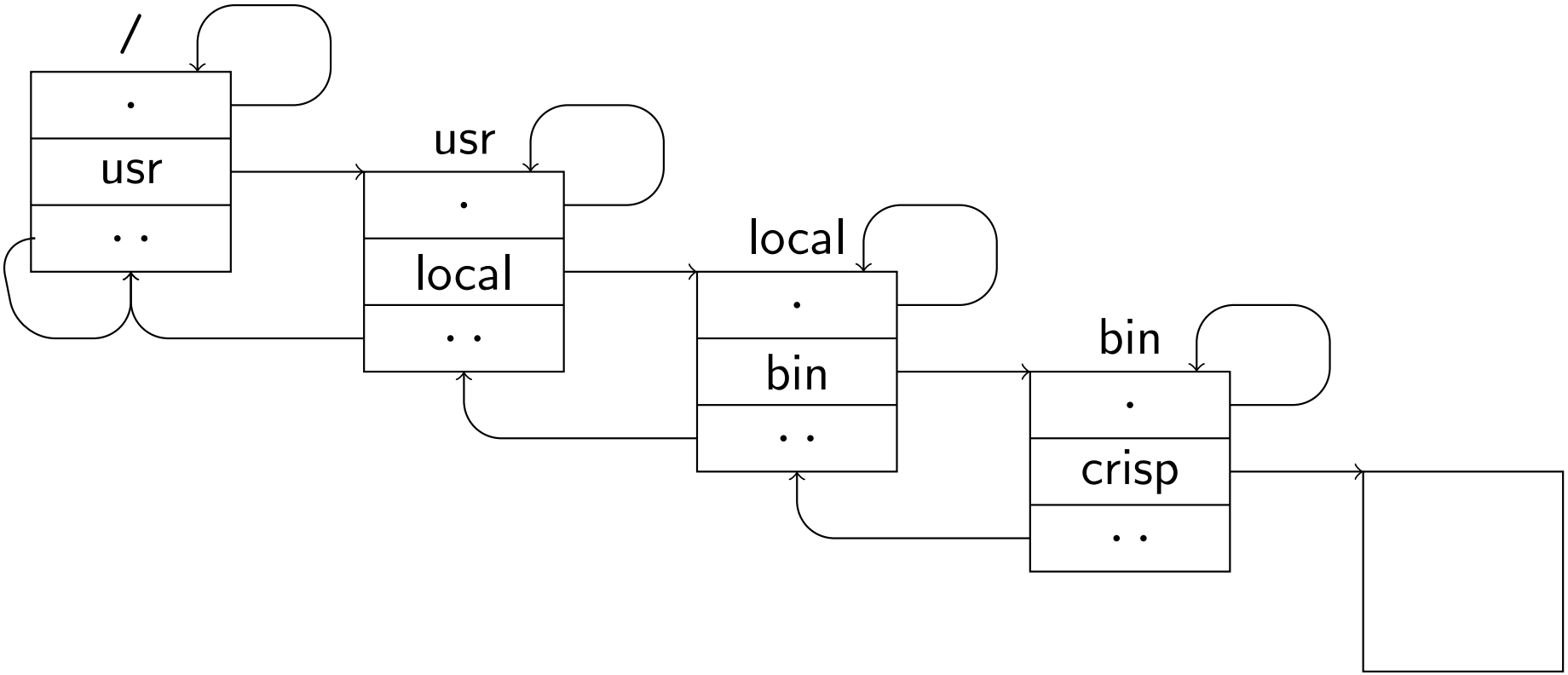
Cykler

- Hvordan kan cykler (i katalog-grafen) opstå?
  - '. .' peger på "forældre-kataloget"
  - Hvad med . . i rodkataloget? Peger på rodkataloget selv!
  - Andre muligheder for cykler?



Cykler

- Hvordan kan cykler (i katalog-grafen) opstå?
  - '..' peger på "forældre-kataloget"
  - Hvad med .. i rodkataloget? Peger på rodkataloget selv!
  - Andre muligheder for cykler? '.' peger på kataloget selv



Metadata

- Hvad er metadata for filer/kataloger?
- Hvor gemmes metadata?

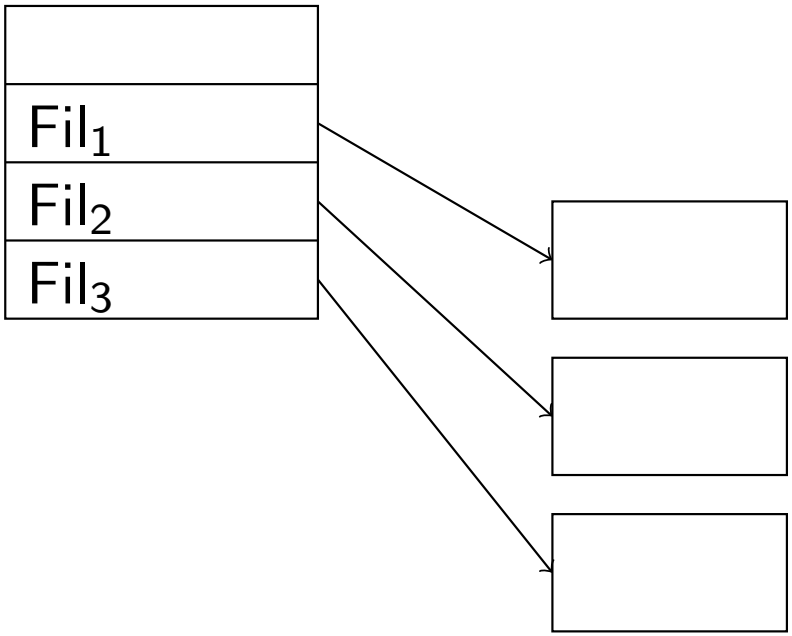
Metadata

- Hvad er metadata for filer/kataloger?
  - Navn
  - Filstørrelse
  - Placering af filen
  - Type
  - Tilgangstider
  - Ejer
  - Rettigheder
  - Udvidede attributter
- Hvor gemmes metadata?

Metadata

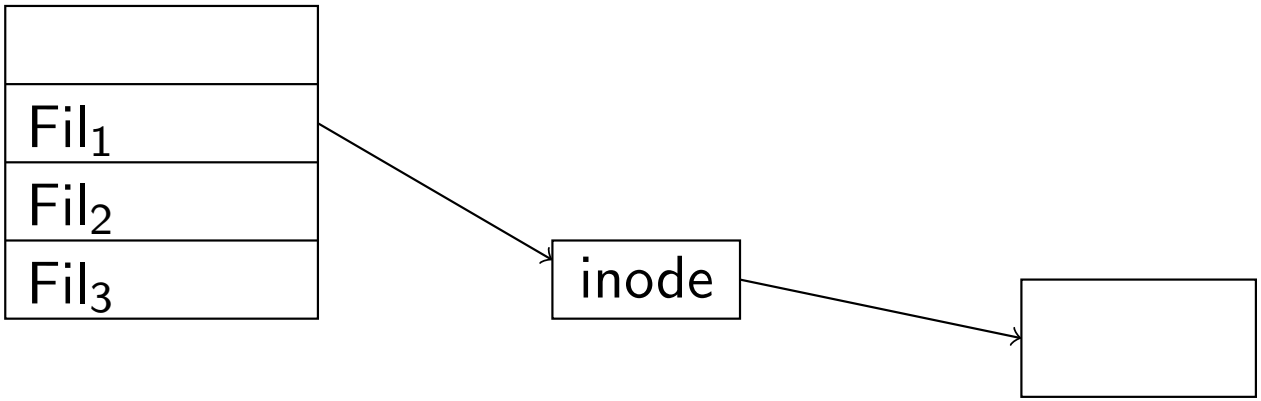
- Hvad er metadata for filer/kataloger?
  - Navn
  - Filstørrelse
  - Placering af filen
  - Type
  - Tilgangstider
  - Ejer
  - Rettigheder
  - Udvidede attributter
- Hvor gemmes metadata?
  - Navne gemmes i kataloger
  - Type, tilgangstider kan gemmes i katalog eller sammen med filens indhold
  - Størrelse og placering kan *ikke* gemmes sammen med filens data. Hvorfor ikke?
  - Udvidede attributter gemmes ofte i særlige dataområder

UNIX: inodes



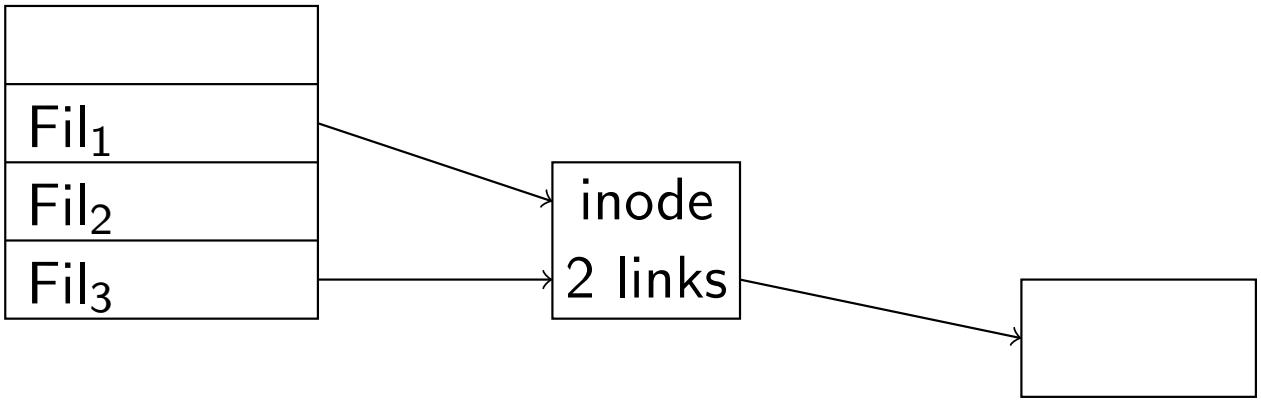
- Fleste metadata gemmes i inode
- Navn gemmes i katalog
- Udvidede attributter gemmes i særligt område

UNIX: inodes



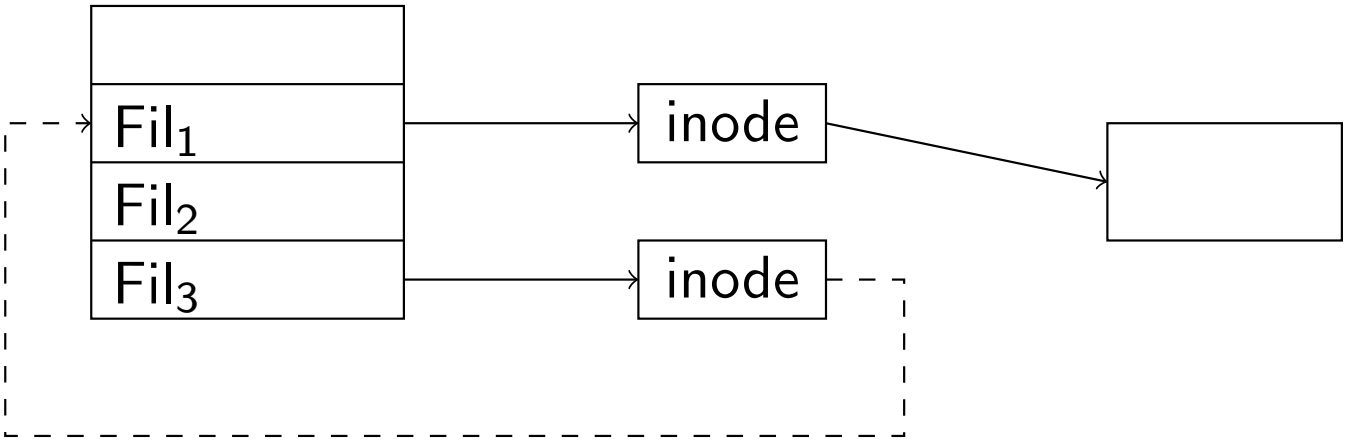
- Fleste metadata gemmes i inode
- Navn gemmes i katalog
- Udvidede attributter gemmes i særligt område

Hard Links



- To katalog-indgange til *samme fil-indhold*
- Vanskeligt at realisere hvis metadata gemmes i kataloget

Symbolic Links

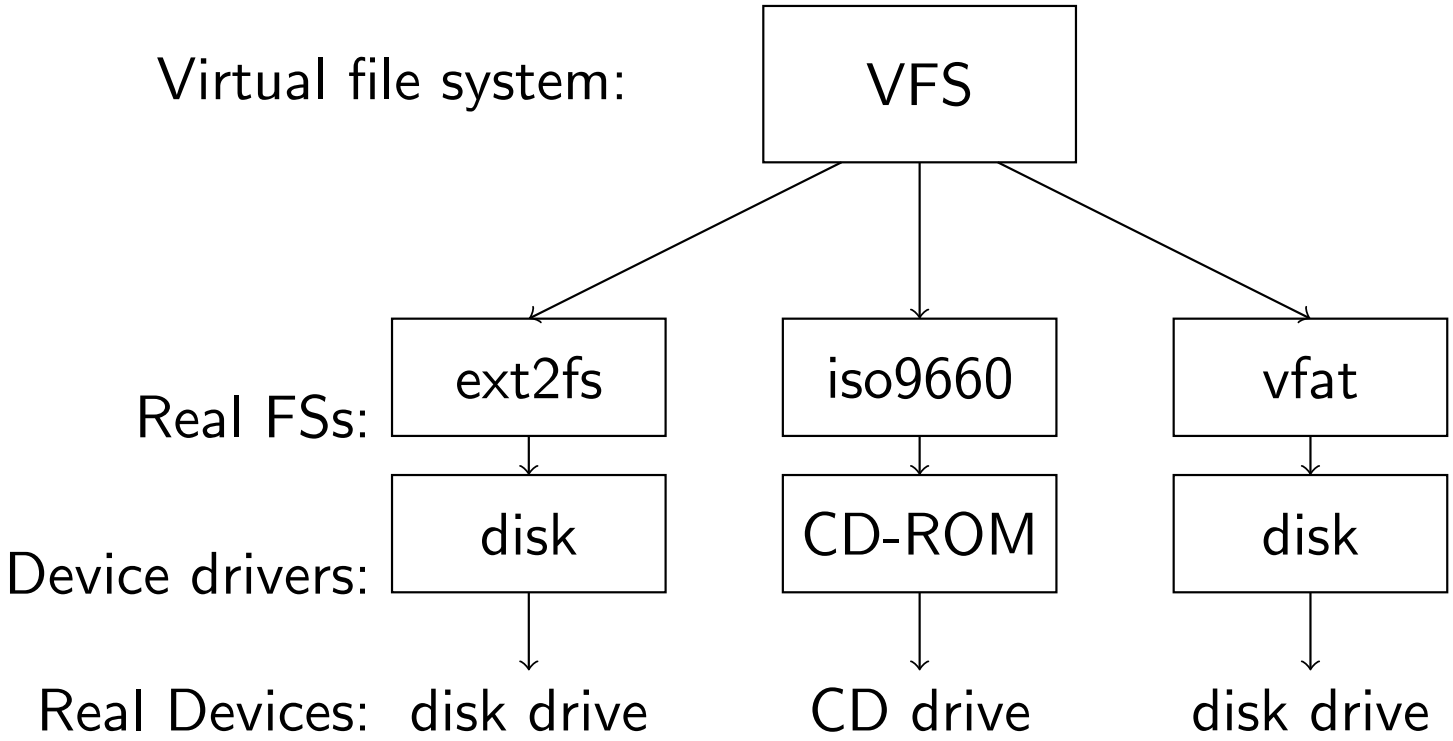


- Speciel attribut markerer filen som et symbolsk link
- Windows' shortcuts implementeret i shellen ikke i filsystemet

Volumes og mount points

- Filsystemer gemmer informationer i volumes
  - Svarer ofte til et medie, fx en CD
  - Et medie kan partitioneres i flere volumes
  - Et volume kan spænde flere medier
- Et volume skal indeholde et rodkatalog
- Styresystmer eksponerer volumes forskelligt:
  - Windows: drevbogstaver
  - Mac OS: ikoner på skrivebordet
  - UNIX: volumes mountes i fælles navnerum

Virtuelle filsystemer



- En sti spænder over forskellige filsystemer
- VFS giver et homogent interface til alle filsystemer
- Filsystemer er ofte implementeret på blokdevices (enheder med et interface baseret på tilgang til lineært nummererede datablokke)
- `open("/mnt/cdrom/dir/foo",...)`

Filtyper

- Filer består af
  - Data
  - Attributter
  - Evt. filnavn
- VMS skelner mellem tre slags filer
  - Sequential
  - Random access
  - Indexed
- UNIX, Windows m.fl.
  - En sekvens af bytes
  - Position i filen (offset i forhold til starten)
  - Struktur er applikationsdefineret
- Windows NT "åndelig" efterfølger til VMS
- Specifikke filtyper vanskeliggør udvikling af filhåndteringsprogrammer
- Kan VMS filtyper bygges ovenpå UNIX'?

Typer af filer

- Behov for at skelne mellem typer af indhold
  - Tekstfiler
  - Binære filer
  - Eksekverbare filer
- Filendelser
  - Integreret del af Windows filhåndtering
  - Konvention på UNIX og Mac OS
  - Mac OS gemmer filtypen som attribut
- Tekstfiler
  - Forskellige tegnsæt
  - Forskellige indkodninger af linieskift
  - DOS bruger speciel End Of Text markering
- Eksekverbare filer
  - Specielle filendelser på Windows
  - Speciel attribut på UNIX og Mac OS

Linux: Fillåse

- Advisory locks — frivillige låse
  - Håndhæves ikke af styresystemet
  - Processer kan blive nægtet en lås, men kan alligevel tilgå den
  - Shared, Exclusive
  - Omfatter hele filen eller dele af filen
- Mandatory locks
  - Ikke en del af POSIX
  - Håndhæves af styresystemet
  - Sættes for en fil via filrettigheder (speciel, ubrugt kombination)
- Systemkald for begge typer låse er identiske
- Når mandatory locks aktiveres bliver advisory locks til mandatory locks

Rettigheder

- Klassisk UNIX
  - Læsning (r), skrivning (w), eksekvering (x) for ejer, gruppe og andre:

    -rw-r------ rrh users
- Access Control Lists (ACLs)
  - Liste der definerer rettigheder for individer og reupper
  - Implementeret i fx Windows med NTFS
  - POSIX ACL understøttes af fx Solaris og Linux (2.6+)

    rrh read write

    users read

Nogle systemkald for filer (UNIX)

- Almindelig brug
  - creat, open, close, truncate, ftruncate, lseek, read, readv, write, writev, dup
- Ejerskab og rettigheder
  - chown, chmod, fchown, fchmod
- Låse
  - flock
- Links
  - lstat, lchown, link, symlink, readlink, unlink
- Andet
  - fcntl, stat, fstat, fsync
- Kataloger
  - getdents, readdir, mkdir, rmdir

Open

- Finder inoden for et givet filnavn
- Cacher inoden
- Opretter en fil-descriptor for processen
- Checker adgangsrettigheder

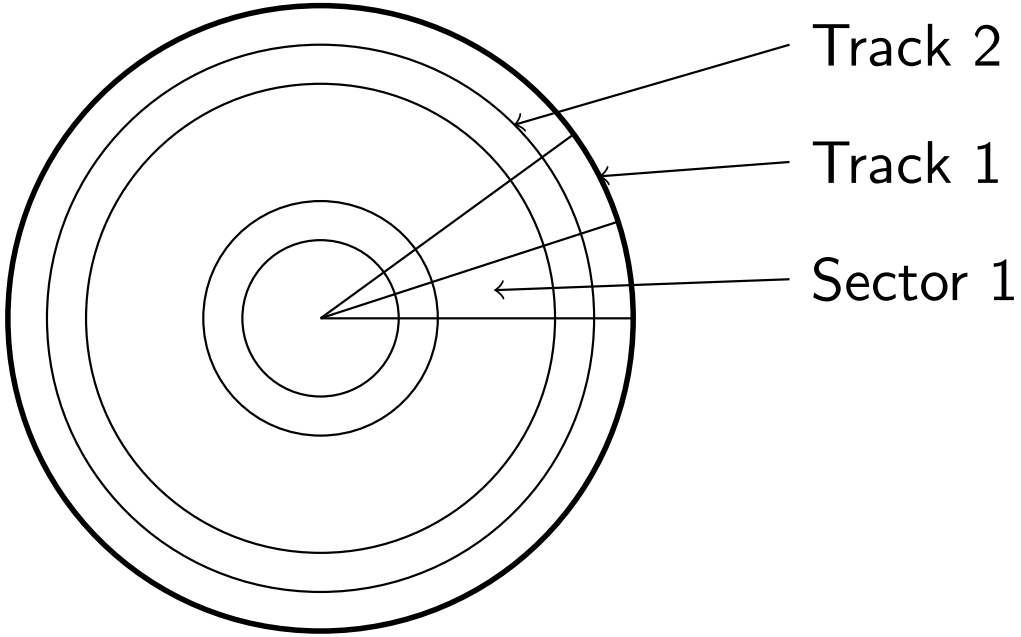
File Descriptor

- Indeholder informationer om den åbne fil
- Hver proces har en tabel over processens åbne filer

```
struct files_struct
  atomic_t  count;
  fd_set    close_on_exec;
  fd_set    open_fds;
  struct file *fd_array[];
```

- Identificeres i processen som et indeks i tabellen

Fysisk Layout



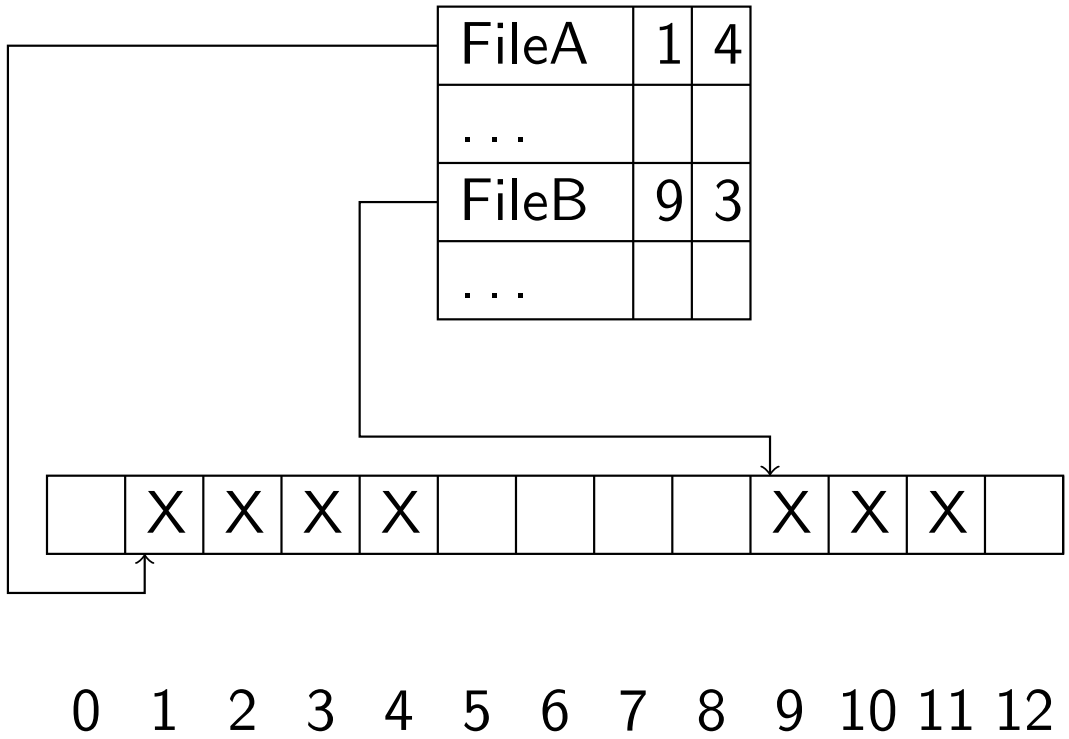

```
class Blockdevice
{
  public BlockDevice(int blockCount);
  public byte[] read(int block);
  public void write(int block, byte[] data);
  public int getBlockCount();
}
```

- Layout afhænger af det faktiske medie
- Typisk med lineært blok-layout
- Kataloger?
- Metadata?
- Sammenhæng mellem blokke og filer?
- Ubrugte blokke?
- Filoperationer
- Hastighed
- Robusthed
- Store filer

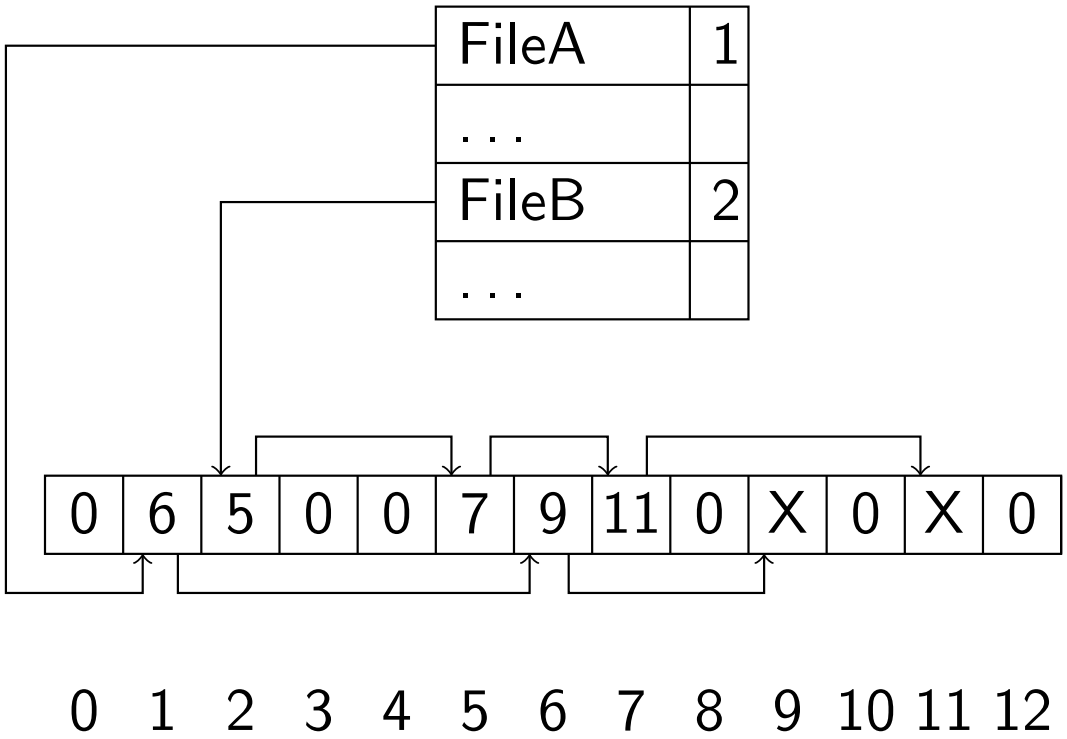
Det ønskede interface

```
class FileSystem
{
  public FileSystem(BlockDevice);

  /* Create a file and return new fd for it */
  public int create(String name);

  /* Open existing file, return fd */
  public int open(String name);

  /* Erase a file */
  public void erase(String name);

  /* Close an open file */
  public void close(int fileDescriptor);

  /* Read byteCount bytes from the file defined by fileDescriptor */
  public byte[] read(int fileDescriptor, int byteCount);

  /* Write data to file defined by fd */
  public byte[] write(int fileDescriptor, byte[] data);

  /* Seek to position in file defined by fileDescriptor */
  public void seek(int fileDescriptor, int position);

  /* Format a block device of the given size */
  public static format(BlockDevice);
}
```

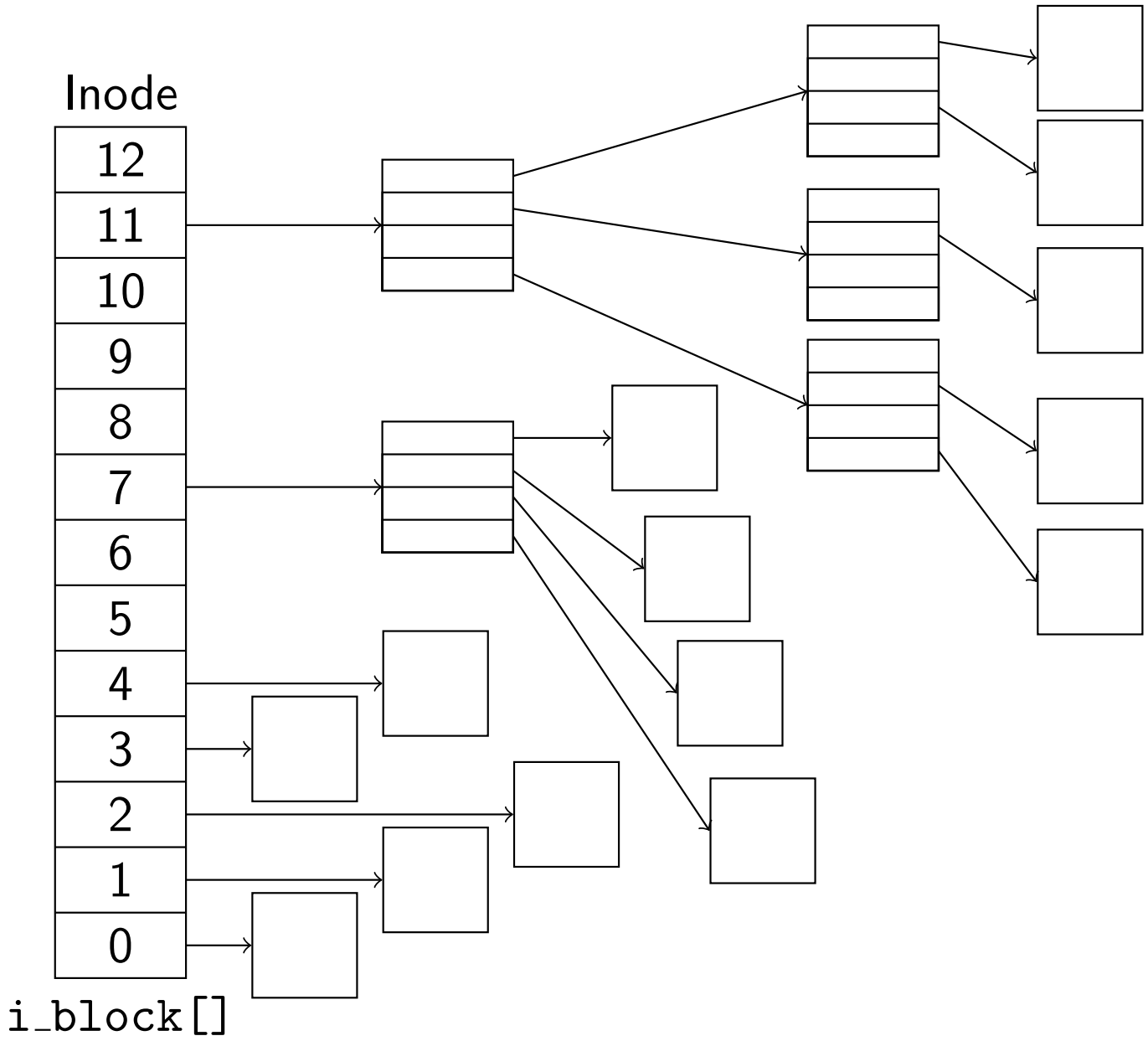
Kontinuert allokering



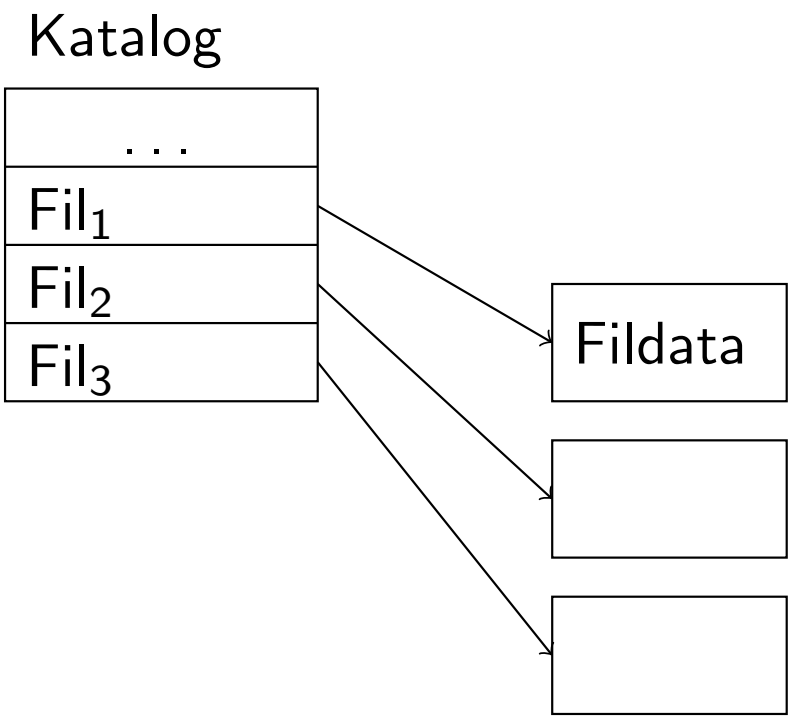
Kædet allokering



Indekseret allokering



Kataloger



- Hvordan gemmes kataloger på disken?
- Kataloger er filer
- Afbilder navne til metadata/inoder
- Afkobler filer og filnavne
- Forskellige formater
- Hvordan allokeres diskblokke til kataloger?
- Hvilke datastrukturer bruges til at gemme katalog-indgange?
- Eksempler: FAT, ext2, NTFS

Opsummering og næste gang

- Filsystemer fra et brugerperspektiv
  - Filsystemer generelt
  - Grundlæggende organisationsprincipper
  - Links
  - Virtuelle filsystemer
  - Grundlæggende egenskaber og operationer
- Næste gang: filsystemer fra et udviklerperspektiv
- Næste gang: processer, tråde og deadlocks