

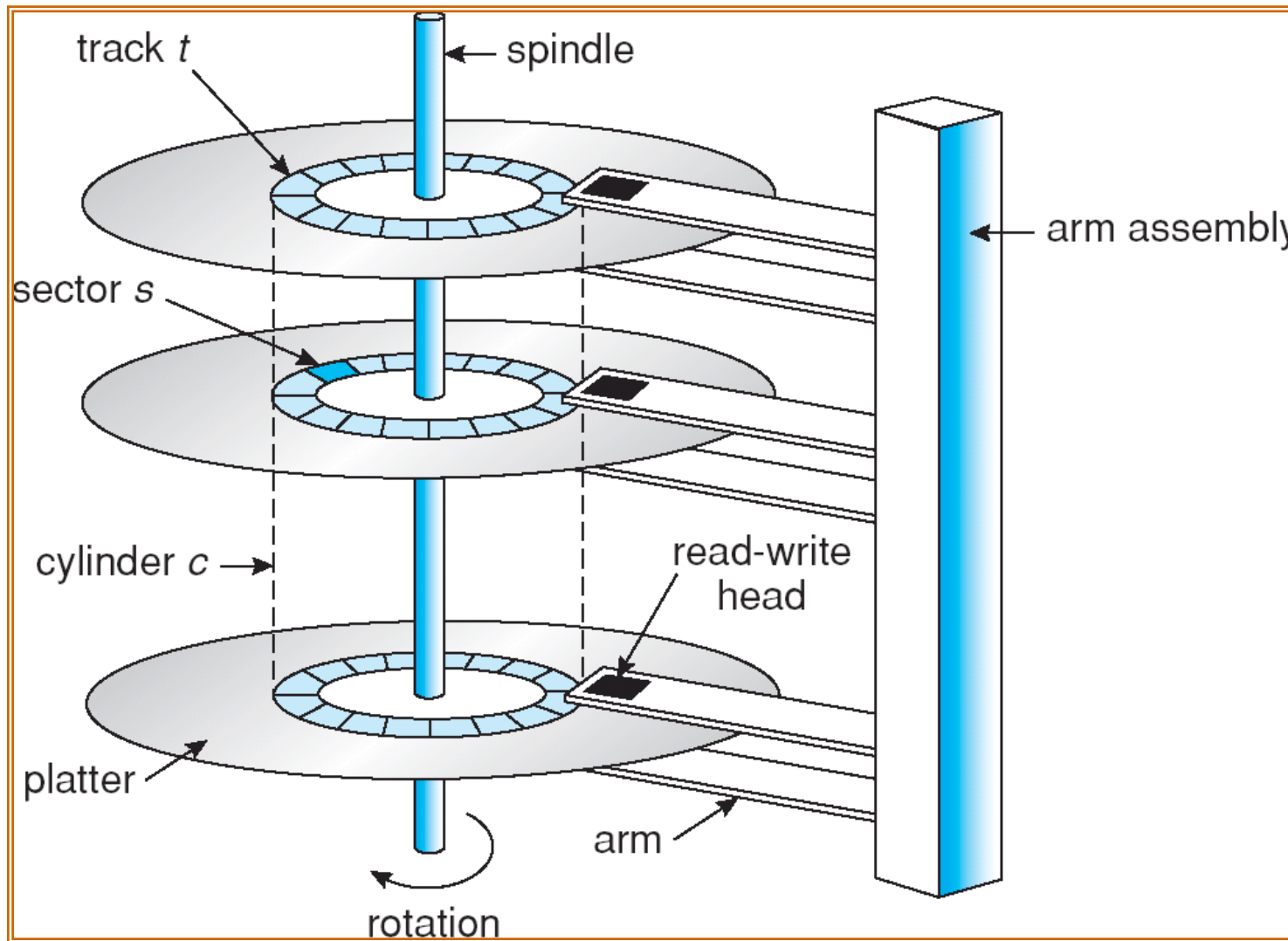
IN1805 I – Operating System Concepten

Hoofdstuk 12: Mass-storage structure

Secondary storage (1)

- voornamelijk disks
- kleinst leesbare eenheid: *sector*
- plaats van een sector volledig bepaald door:
 - drive nummer (diskpack)
 - de plaat (head)
 - de track
 - de sector
- *cylinder* : tracks die op verschillende platen vertikaal onder elkaar liggen
- Disk voor te stellen als een ééndimensionale array van sectoren, met volgorde van nummering:
 - eerst sectoren van cylinder 0, head 0
 - daarna die van cylinder 0, head 1, etc.

Secondary storage (1a)



Secondary storage (2)

- Een proces doet I/O-requests op basis van logisch sectornummer (512 byte block)
- het file-organisatie module vertaalt logisch sectornummer naar cylinder, track en sector adres
- Op I/O-control niveau (device driver) worden de requests uitgevoerd.

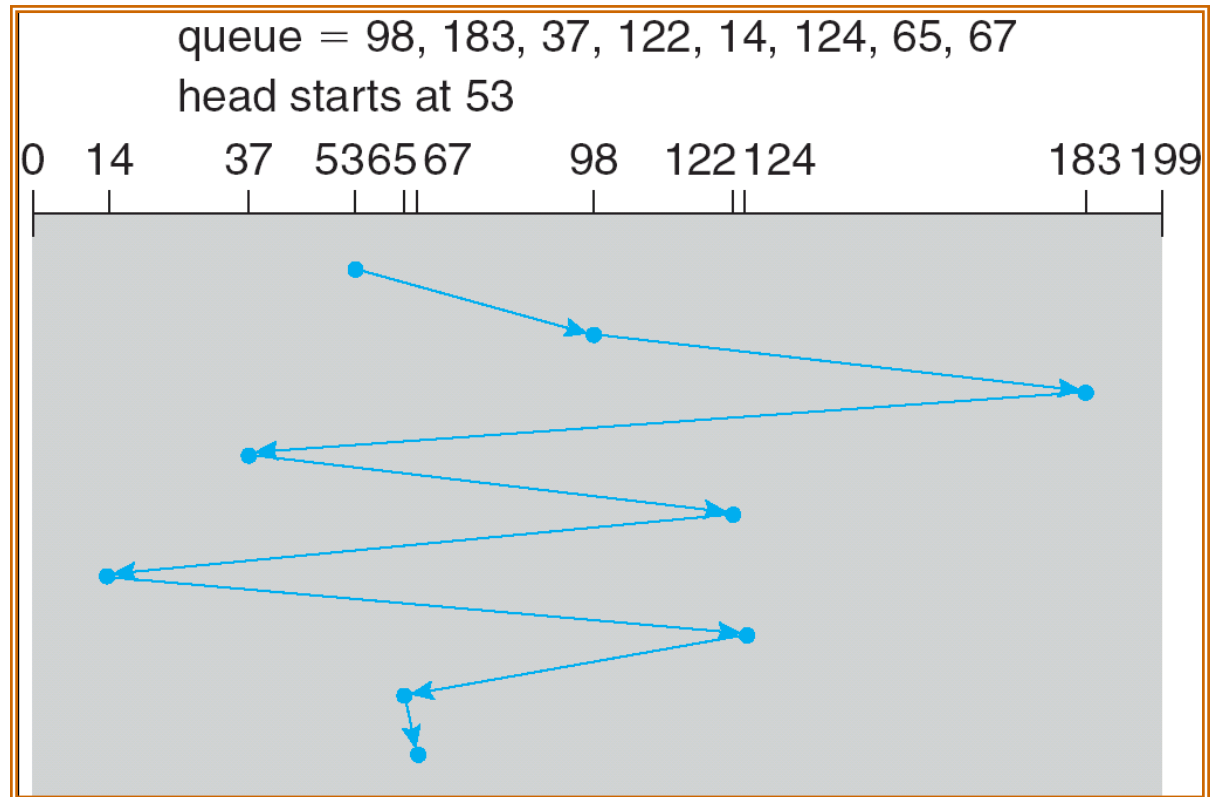
Disk-head scheduling (1)

- Tijd tussen het starten van de I/O-operatie en het einde van de I/O operatie , bestaat uit:
 - seektijd (t.b.v. het bewegen van de kop naar de goede track)
b.v. 10 ms
 - latencytijd (tot goede sector onder de kop)
b.v. 8 ms
 - transfertijd (het lezen of schrijven zelf)
b.v. 1 ms
- Seektijd evenredig met het aantal 'over te steken' tracks
- Bij meer dan één wachtende I/O request, seektijd te bekorten door goede disk-head scheduling

Disk-head scheduling Algoritmen (1)

- FCFS (First Come First served)

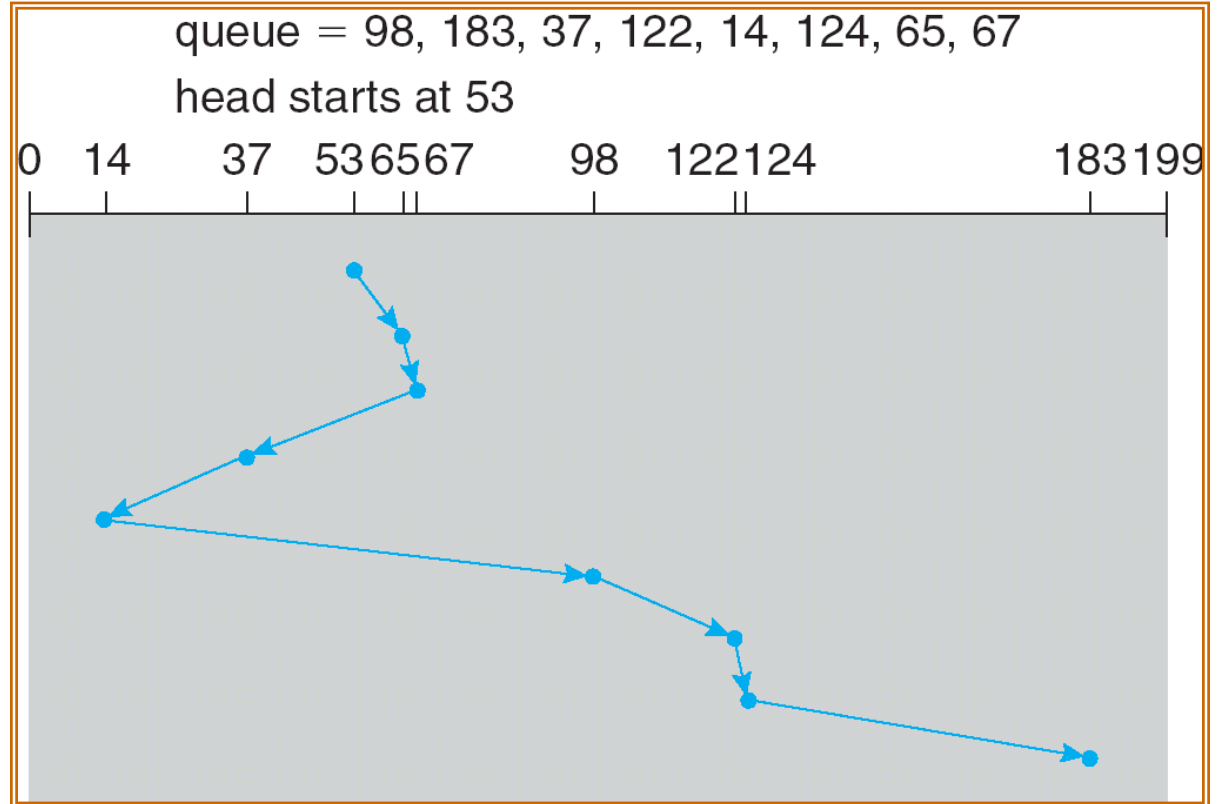
- eenvoudig
- geen starvation
- seektijden niet optimaal



Disk-head scheduling Algoritmen (2)

- SSTF (Shortest Seek Time First)

- gunstiger seektijden
- starvation mogelijk



Disk-head scheduling Algoritmen (3)

- SCAN (lift algoritme: op en neer van links naar rechts)

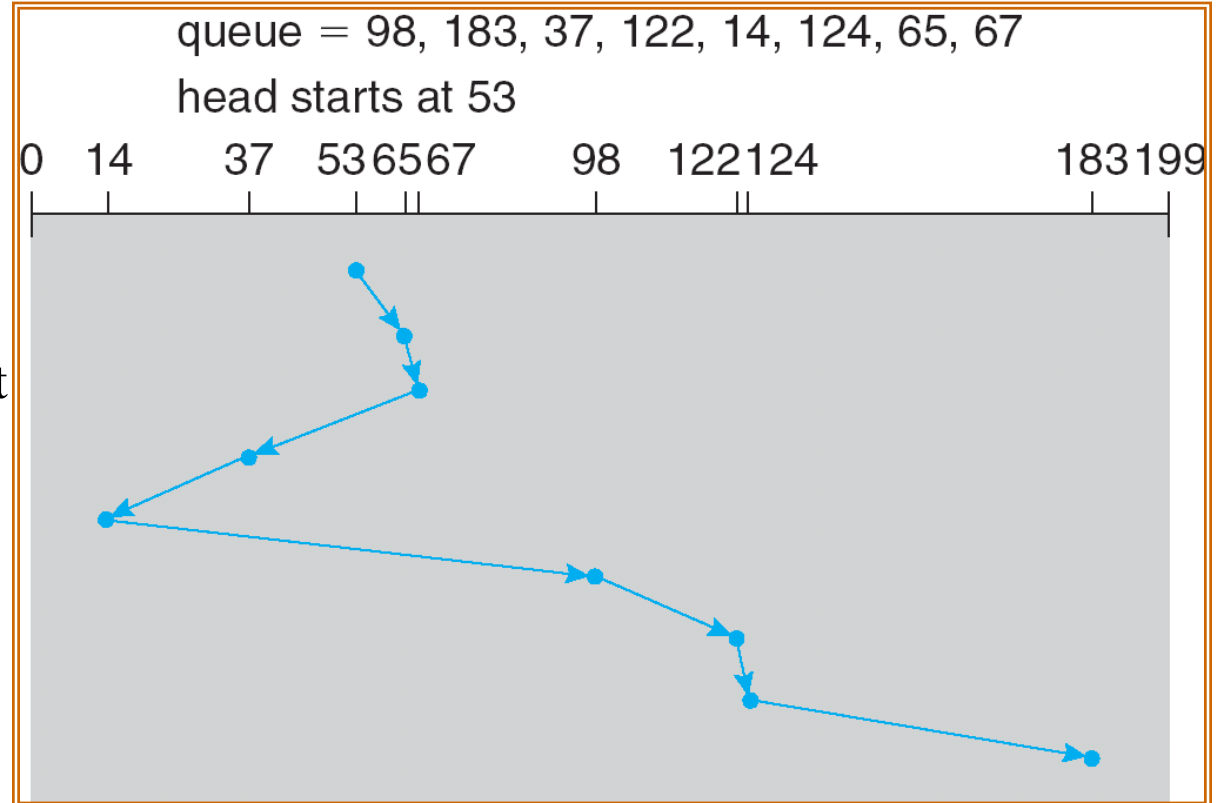
- OK seektijden
- geen starvation

- LOOK

- keer terug bij max/min request

- SCAN/LOOK-C

- Circulair
- uniforme access tijd



Disk head scheduling (2)

keuze van een algoritme

- Als zelden meer dan 1 request in de queue, dan is ieder algoritme goed.
- patroon van de requests heeft ook te maken met
 - single/multi user systeem
 - file allocation methode
 - positie van de directory
 - positie van index tabel
- SSTF veel gebruikt
- Bij zware belasting SCAN of SCAN-C beter

Waarom ?

IN1805 I – Operating System Concepten

Hoofdstuk 11: File system implementation

File systeem implementatie

introductie

- Files worden opgeslagen op permanent geheugen (disk, tape, ...)
- alleen disks worden hier besproken
- model
 - I/O van main memory naar disk in de vorm van blokken van vaste grootte.
 - 1 blok op disk is 1 of meer sectoren (van b.v. 512 bytes)
 - een blok heeft een uniek blok-adres
 - als het blok-adres bekend is, dan is het blok direct te benaderen.
 - een blok is op dezelfde plaats terug te schrijven

File systeem implementatie organisatie

gelaagde structuur:

- applicatie programma
gebruikt system calls (evt. via een subroutine library)
- logisch file systeem
gebruikt directory om file te vinden
- file organisatie
vertaalt logisch blok-adres naar fysiek blok-adres
- Basis File Systeem
geeft lees- en schrijfoopdrachten voor blokken
- I/O control
bevat device drivers en interrupt control routines
- devices
de controllers en de schijven

Structuren (1)

overzicht

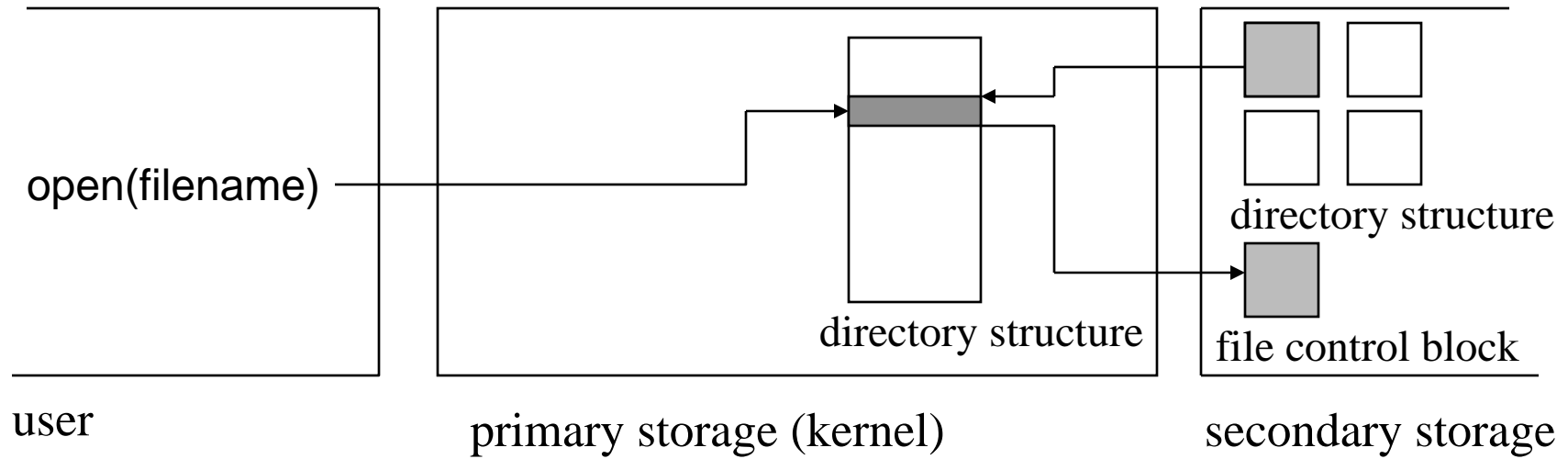
Op disk

- Boot control block (UNIX: *boot block*)
- Partition Control Block (UNIX: *superblock*)
- Directory structure
- File Control Block (UNIX: *inode*)
- data blokken

In primary storage

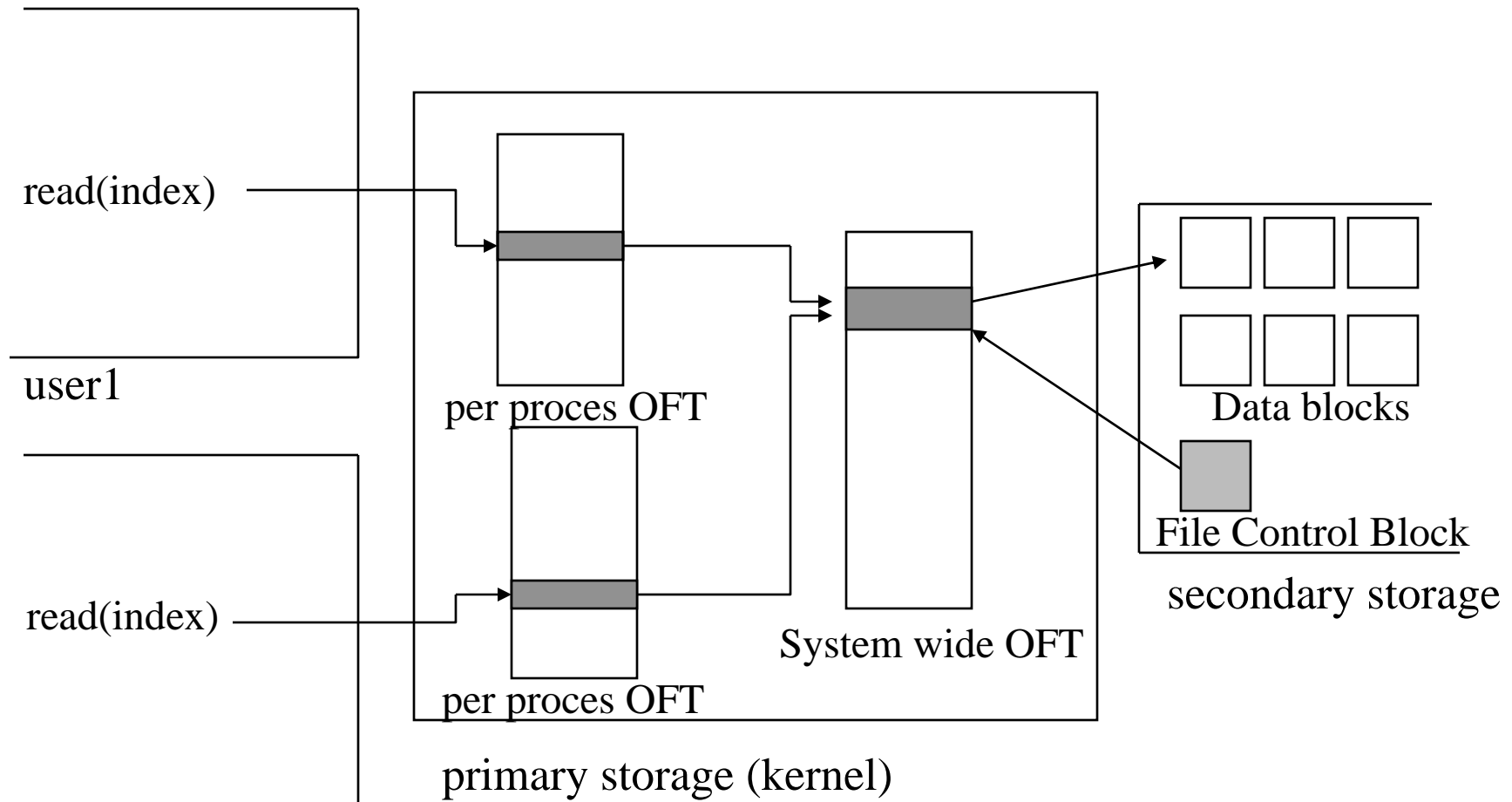
- partitie tabel
- directory informatie
- system-wide Open File Table
 - bevat FCB van iedere open file
- per-process Open File Table
 - bevat o.a. current position pointer

structuren (2)



Het openen van een file

structuren (3)

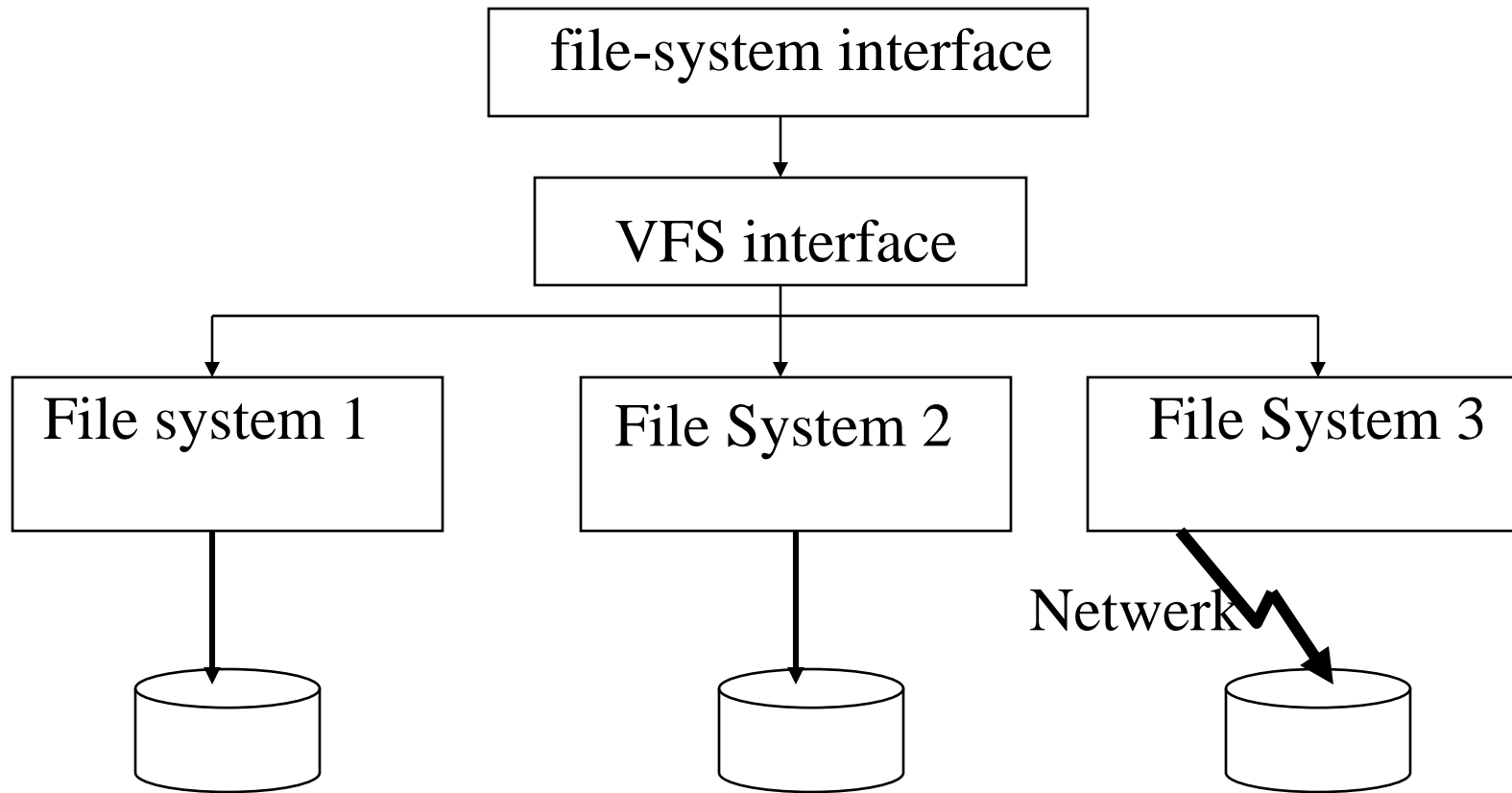


Het (shared) lezen van een file door twee processen

Virtual File Systems

- Een VFS maakt het mogelijk verschillende file systemen in één Operating Systeem te ondersteunen.
 - b.v. in linux: ffs, extfs, ext2fs of procfs
 - b.v. in Windows: FAT, NTFS
- Een VFS biedt één interface naar de applicatie
- verschillen in file systemen worden opgevangen door de implementatie van het VFS

VFS



PAUZE

Toekennen van schijfruimte aan files overzicht

Allocatie methoden:

- aaneengesloten (contiguous allocation)
- geketende blokken (linked allocation)
- geïndexeerde blokken (indexed allocation)

contiguous allocation (1)

aaneengesloten blokken

voordelen:

- minimum aantal seeks nodig
- eenvoudige administratie
- geschikt voor sequentiële en directe toegang

nadelen

- externe fragmentatie : dynamic allocation problem; soms compaction nodig
- interne fragmentatie: grootte van tevoren schatten is vaak moeilijk voor een gebruiker

contiguous allocation (2)

variant:

- primaire allocatie van een aantal aaneengesloten blokken
- wanneer primaire allocatie niet voldoende, dan een aantal extra gebieden van aaneengesloten blokken (extents) mogelijk.
- extents hoeven niet te grenzen aan de primaire allocatie of aan elkaar
 - (voorbeeld IBM: MVS grootte van primary en van extents op te geven door gebruiker. Maximaal 15 extents mogelijk)

linked allocation (1)

geketende lijst van even grote blokken verspreid over de disk
voordelen

- geen externe fragmentatie
- beperkte interne fragmentatie
- alle files kunnen groeien

nadelen

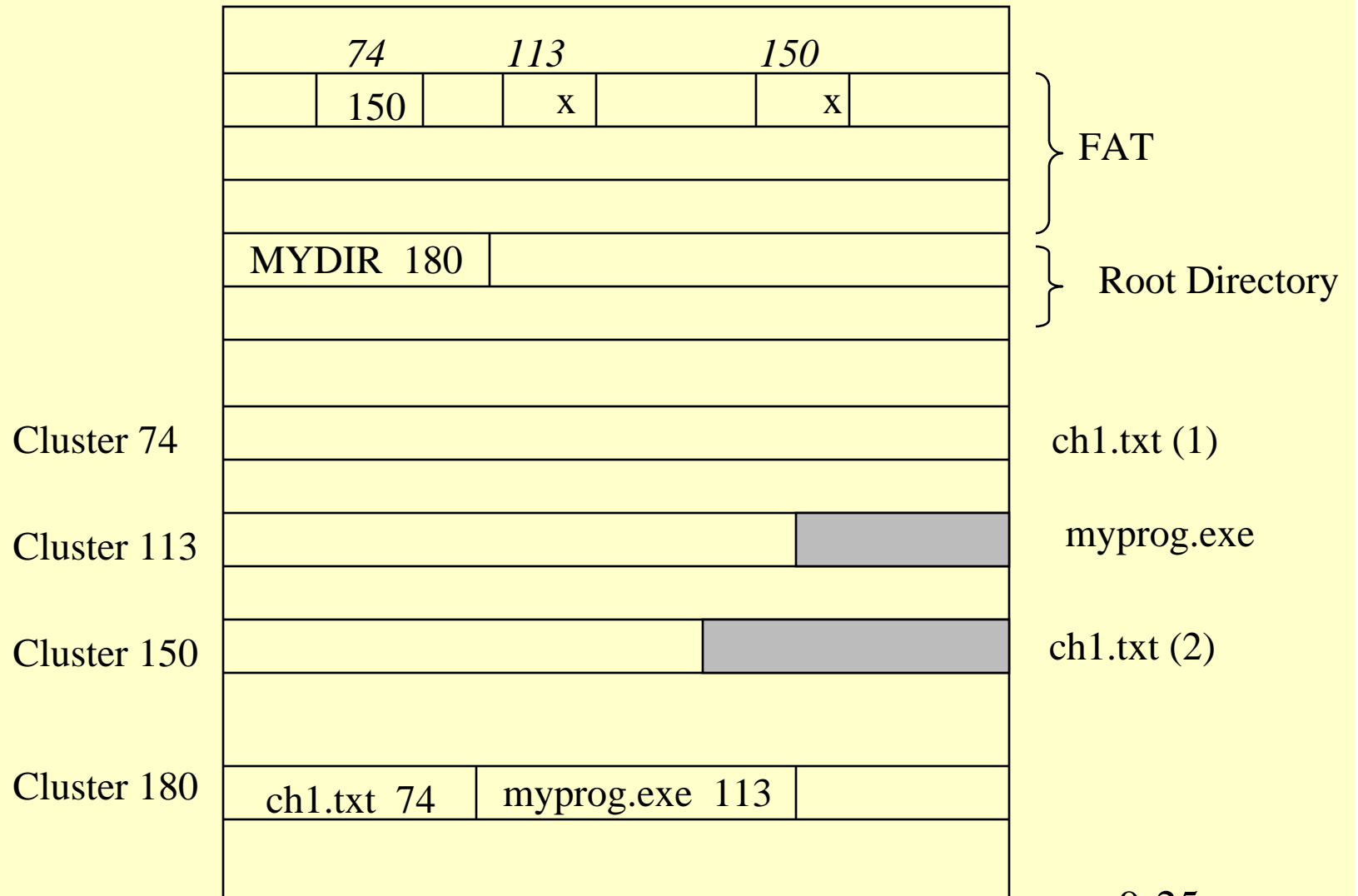
- alleen efficiënt voor sequentiele toegang
- pointers kosten ruimte
- veel seeks nodig (clusters)
- gevoelig voor verminking (of verlies) van pointers

linked allocation (2)

variaties

- dubbel gelinkte lijst i.p.v. enkelvoudig gelinkt
- MS-DOS File Allocation Table (FAT)
 - pointer naar eerste blok bevindt zich in de directory,
 - pointer naar het volgende blok bevindt zich niet in een blok, maar in de FAT

MSDOS disk layout, FAT table voorbeeld



indexed allocation (1)

Files in blokken verspreid over de schijf ; pointers naar de blokken worden per file bewaard in een index blok

voordelen:

- weinig seeks nodig (bij direct access)
- geschikt voor sequentiële en directe toegang
- alle files kunnen groeien

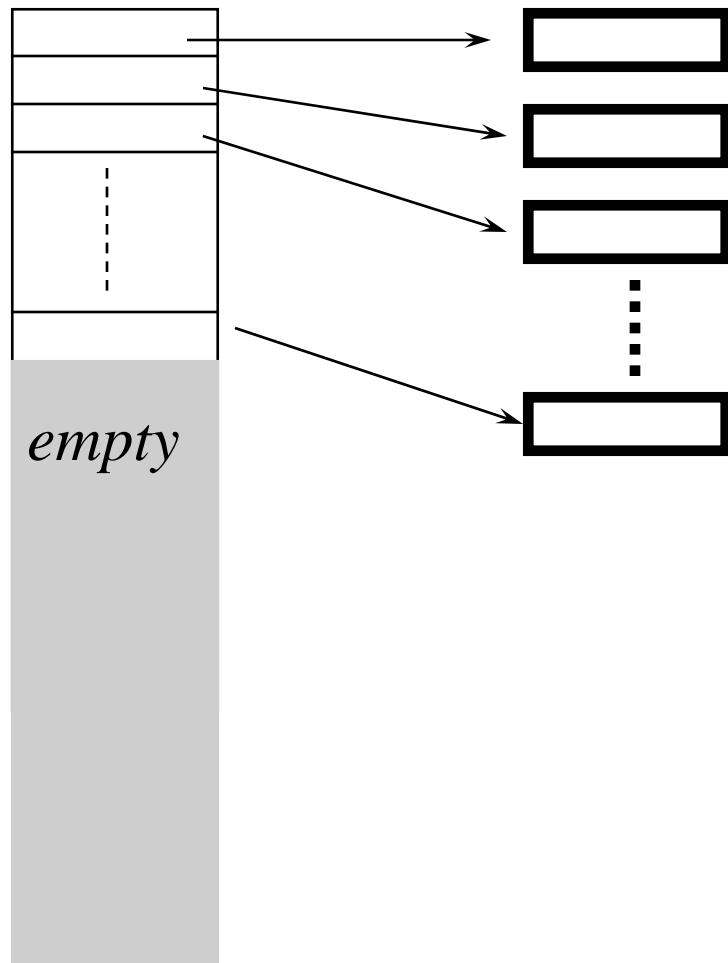
nadelen:

- overhead (ruimteverlies) t.g.v. pointers
- Index blok vaak voor grootste deel leeg (veel kleine files)

Indexed Allocation: basis (met 1 index block)

index block

data blocks

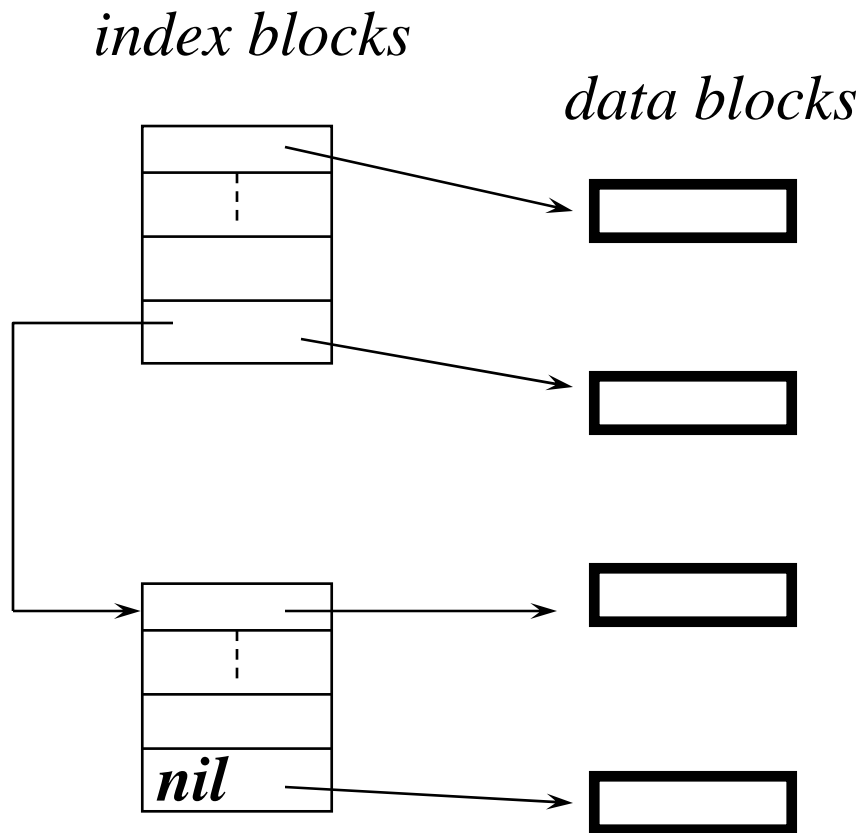


Indexed allocation (2)

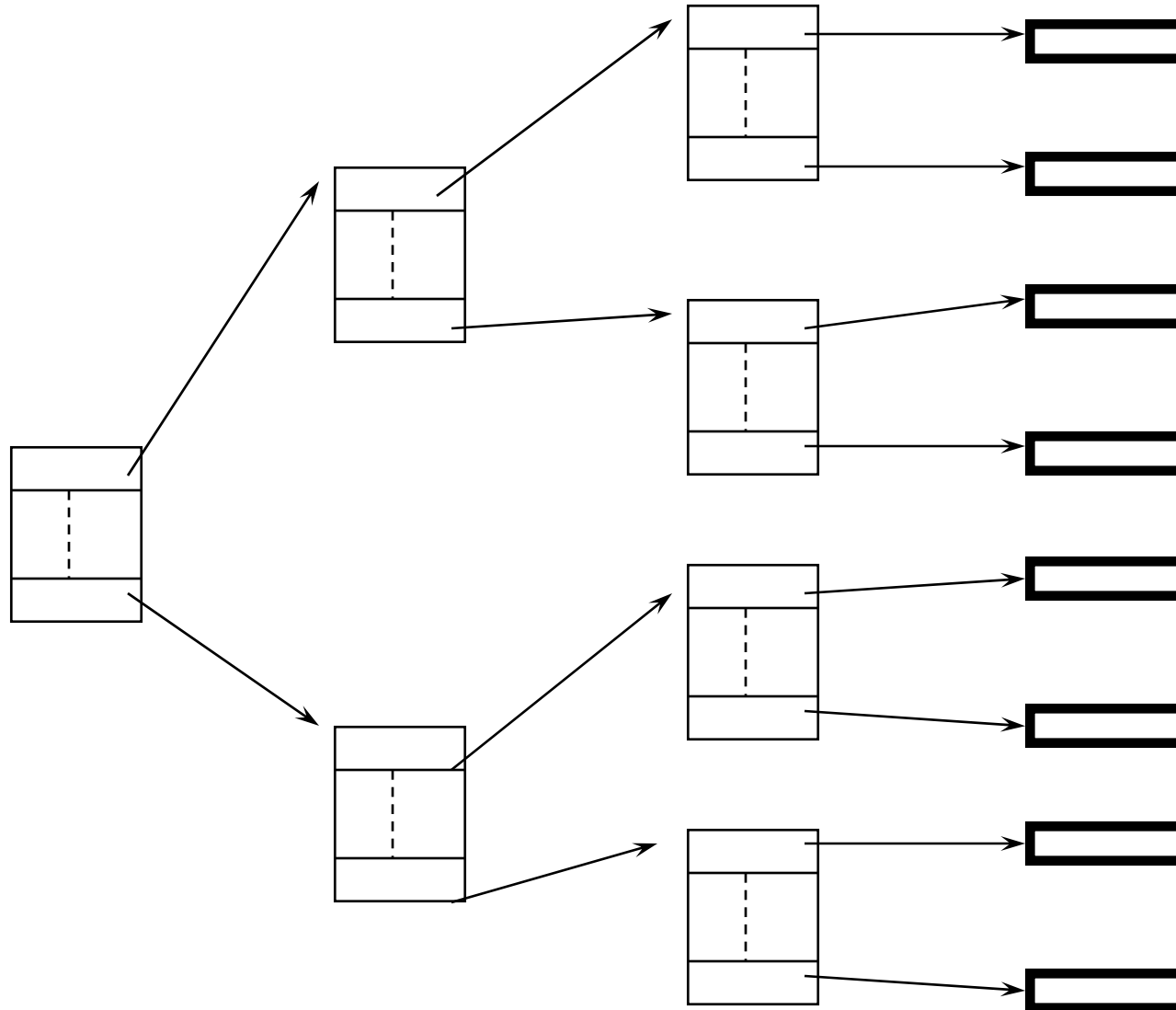
varianten met oplossingen voor de grootte van het index blok

- (kleinere) geketende indexblokken (*linked scheme*)
- multilevel index
- combinatie (vast minimum + multilevel hiërarchie)

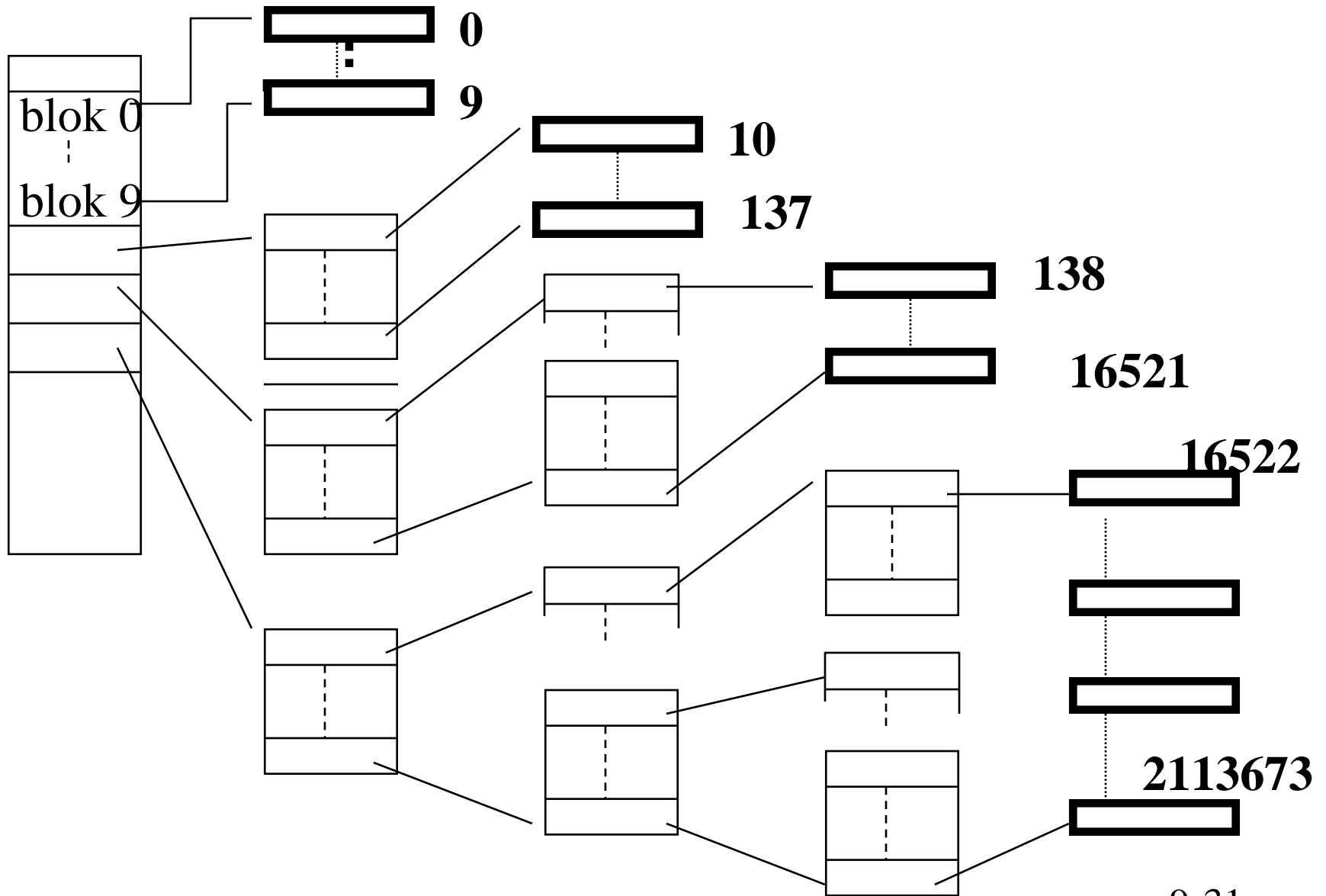
Indexed Allocation: met linked index blocks



Indexed allocation: met multilevel index blocks voorbeeld: 3-voudige indirectie



Indexed allocation : vast aantal indexen + multilevel
voorbeeld met 128 pointers per index block



Vrije ruimte beheer

- bitmap (bitvector)
 - Ieder geheugen blok wordt gerepresenteerd door 1 bit (0 als bezet, 1 als vrij)
 - voordeel: snel te doorzoeken: aaneengesloten vrije ruimten snel te herkennen
- linked list
 - vrije blokken vormen een keten, verbonden door pointers in de blokken
 - voordeel: kost geen extra ruimte