erasmus

HOGESCHOOL BRUSSEL

IT Essentials

Deel III: Operating Systems

4: Filesystems

INHOUD

- Introductie
- Files
- Directories
- File system implementatie
- File system beheer en optimalisatie
- Overzicht filesystemen

erasmus
HOGESCHOOL BRUSSEL

INTRODUCTIE

- **Non-volatile storage** (op HDD) werkt met read en write block
- Zeer onhandig voor gebruiker/programmeur
- **Abstractie** vereist door het OS
- Files en directories
- Files zijn analoog aan process address spaces, maar moeten blijven bestaan als process wordt afgesloten
- **File system** is de manier waarop een OS deze abstractie zal uitvoeren

FILES

- Benoeming
- Type
- Toegang
- Attributen
- Operaties

erasmus
HOGESCHOOL BRUSSEL

FILES

- Benoeming
- van 8 tot 255 karakters lang
- Case-sensitive (Posix) of niet (Windows)
- Extensies:
  - In Windows 3 karakters (meestal) en gebruikt om juiste applicatie te activeren
  - In Posix vrije lengte en soms dubbel (vb .tar.gz). Niet noodzakelijk

erasmus HOGESCHOOL BRUSSEL

FILES

- Types
- Regular files
  - ASCII: inhoud gewoon leesbaar in teksteditor
  - Binary: onderhevig aan vaste structuur (bijvoorbeeld magic number als start van een executable file)
- Directories
  - Systeembestanden om de filestructuur te handhaven

erasmus
HOGESCHOOL BRUSSEL

FILES

- Character files (posix only)
  - Bestanden gebruikt om seriële I/O devices weer te geven
- Block files (posix only)
  - Bestanden gebruikt om storage devices weer te geven

erasmus
HOGESCHOOL BRUSSEL

FILES

- Access
- Sequentieel
- Random-access
  - Let op met cyclische access
  - Noodzakelijk voor moderne applicaties
  - 2 mogelijke manieren:
    - Lees vanaf deze positie
    - Verander de positie (via (l)seek) en begin (sequentieel) van deze positie te lezen

FILES

- Attributen
- Naast filename en datum nog andere eigenschappen van een bestand
- Ook metadata genoemd
- Variëren sterk van OS tot OS

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

FILES

- Operaties
- zie ook system calls
  - *create* en *delete*
  - *open* en *close*: noodzakelijk voor elke operatie op een file
  - *read* en *write*: meestal van de huidige positie in de file (zie seek)
  - *append*: speciale write, enkel op het eind
  - *seek*: verander positie in de file
  - *get* en *set attributes*
  - *Rename*: soms via create, copy en delete

erasmus
HOGESCHOOL BRUSSEL

DIRECTORIES

- Speciale files gebruikt om de structuur te bewaren
- Helemaal bovenaan: de root directory
- Single-level versus multi-level: hiërarchische boomstructuur
- Nood aan bepalen van locatie binnen de boomstructuur

erasmus
HOGESCHOOL BRUSSEL

DIRECTORIES

- Pathname
  - **Absoluut**: onafhankelijk van de locatie in de boomstructuur
    - Windows: C:\users\gebruikerx
    - Posix: /users/gebruikerx
  - **Relatief**: afhankelijk van de locatie in de boomstructuur
    - Locatie is de **working directory**
    - Elk process eigen working directory
    - soms vereist indien absolute locatie van bestanden niet gekend is, maar de relatieve wel (bv. bij programma's)
    - Werken met .. en .

erasmus HOGESCHOOL BRUSSEL

DIRECTORIES

- Operaties
- zie ook system calls
  - *create* en *delete*
  - *opendir* en *closedir*: noodzakelijk voor elke operatie op een file
  - *readdir*
  - *rename*
  - *link* en *unlink*: files worden getoond in meerdere directories door het verhogen van de i-node van de file

erasmus HOGESCHOOL BRUSSEL

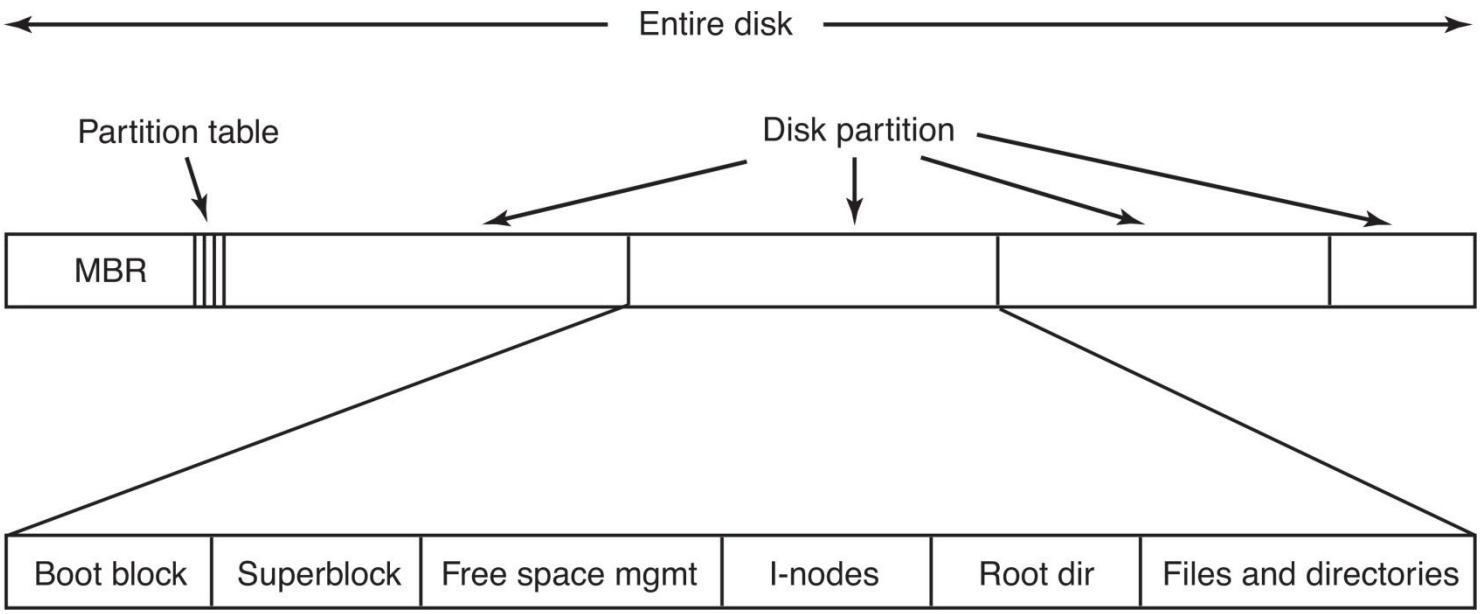
File-system implementatie

- File-system layout
- Disks kunnen opgedeeld zijn in meerdere partities (fase 2 format)
- Sector 0 op de schijf is de **MBR**
  - BIOS of UEFI leest de MBR bij het opstarten
  - Op het eind van de MBR: partition table
  - 1 partitie is active
- 1ste block van de active partition: boot block
- Wordt uitgevoerd en laadt het OS
- Meer en meer vervangen door **GPT** (Guid Partition table): geen partitie beperkingen meer in aantal (4) en grootte (2TB)

erasmus HOGESCHOOL BRUSSEL

File-system implementatie

- File-system layout
- Vaak ook superblock met extra info over het file-systeem



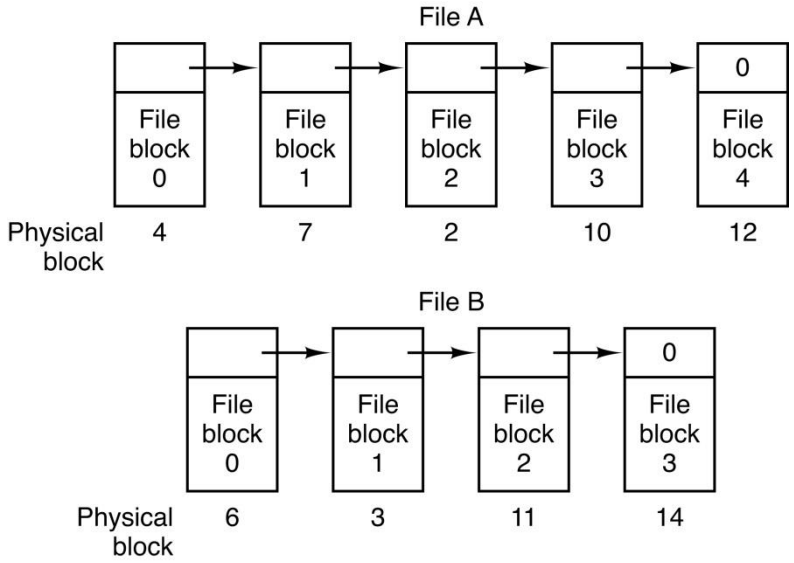
erasmus HOGESCHOOL BRUSSEL

File-system implementatie

- File-allocation
  - Welke blocks worden door welke file gebruikt?
- Contiguous allocation
  - Alle blocks van een file achter elkaar
  - Eenvoudig en snel
  - Veroorzaakt gaten bij het deleten van files en uiteindelijk fragmentatie
  - Opvullen van de gaten enkel mogelijk als de grootte van de te schrijven file op voorhand is gekend
  - Zeer moeilijk werkbaar bij vele userfiles zoals .docx
  - Nog steeds gebruikt bij CD-ROM's
- Linked-list allocation
  - Eerste woord van een block wordt gebruikt om te wijzen (pointer) naar de volgende block gebruikt door de file

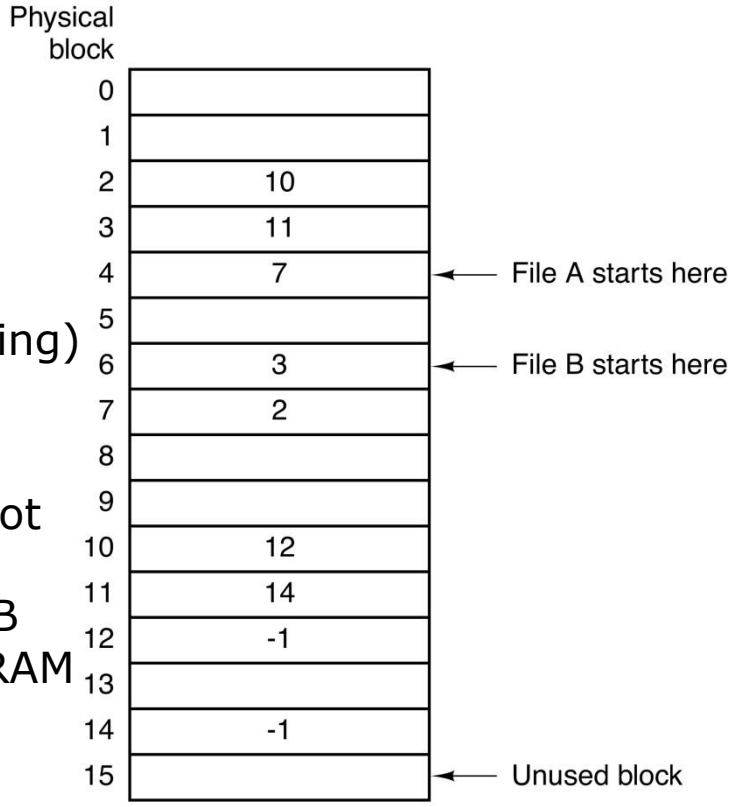
File-system implementatie

- Enkel het adres van de eerste blok dient opgeslagen te worden
- Random-access is zeer traag: alle blocks vanaf de eerste moeten doorlopen worden om het adres van de gezochte data te vinden
- Veel programma's schrijven data met grootte van machten van 2: niet meer mogelijk door verlies 1ste woord



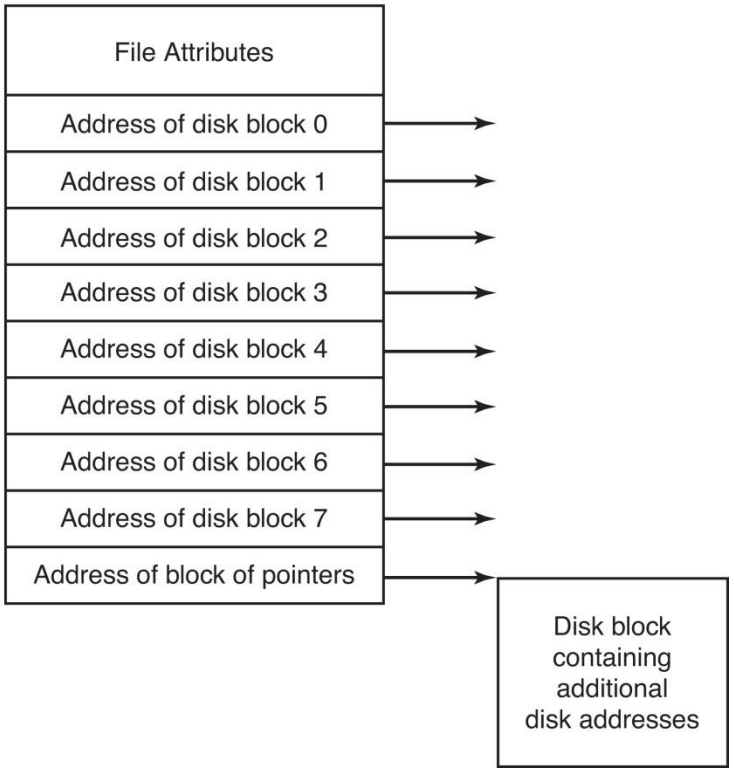
File-system implementatie

- Linked-list allocation met tabel in geheugen
  - Pointer wordt in een tabel in het geheugen geplaatst
  - Voorbeeld:
    file A: begint bij 4
    file B: begint bij 6
  - -1 wordt gebruikt om laatste block weer te geven (ongeldige verwijzing)
  - **File Allocation Table (FAT)**
  - Tabel kan zeer groot worden: voor 1TB-schijf met 1KB blocks: 2,4-3 GB RAM nodig!



File-system implementatie

- I-nodes
  - Voor elke file een I(ndex)-node die de attributen en de gebruikte blocks bijhoudt
  - Enkel in het geheugen wanneer de file open is
  - Structuur I-node:



File-system implementatie

- Shared files
  - Extra referentie naar een file die maar 1 keer wordt opgeslagen
  - Hard-links: verwijzing in beide directories naar I-node die de blocks bijhoudt
  - Symbolic links: extra file wordt aangemaakt met verwijzing naar de oorspronkelijke file

erasmus HOGESCHOOL BRUSSEL

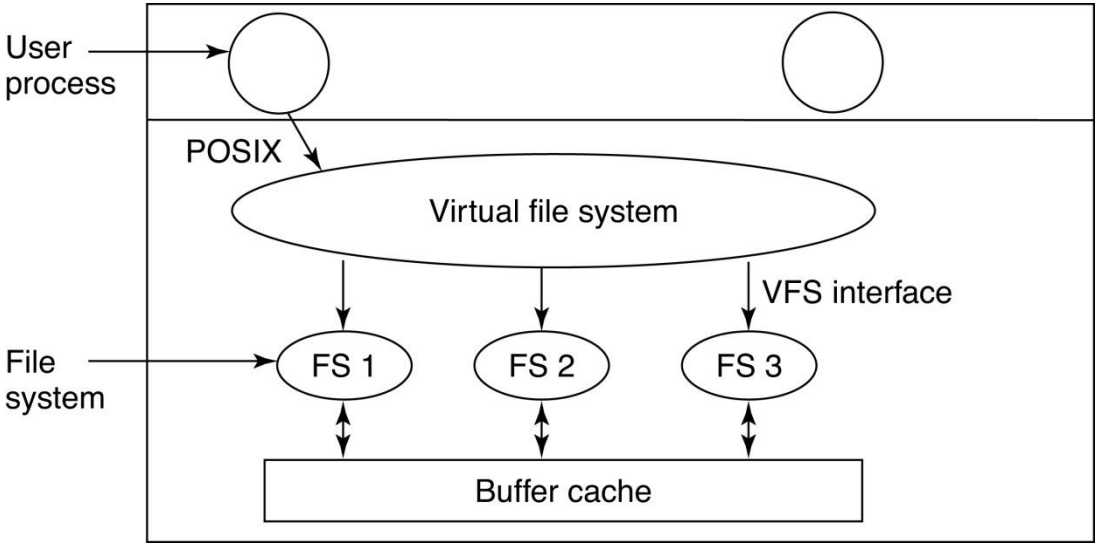
File-system implementatie

- Journaling file-systems
  - NTFS, EXT3, ReiserFS en OSX (optioneel)
  - Doel: stabiliteit verhogen
  - Voorbeeld: file verwijderen
    - File uit de directory verwijderen
    - De I-node releasen voor nieuw gebruik
    - De blocks releasen voor nieuw gebruik
  - Wat bij een systeemcrash zonder dat alle 3 zijn uitgevoerd, onafhankelijk van de volgorde van uitvoering?
  - Journal: log waarin de operaties eerst worden neergeschreven alvorens deze uit te voeren

File-system implementatie

– Virtuele file-systemen (VFS)

- Vaak meerdere verschillende file-systemen gebruikt door 1 OS
- Windows: verschillende Drive-letters
- Posix: ook dit abstraheren naar 1 boomstructuur: **VFS**



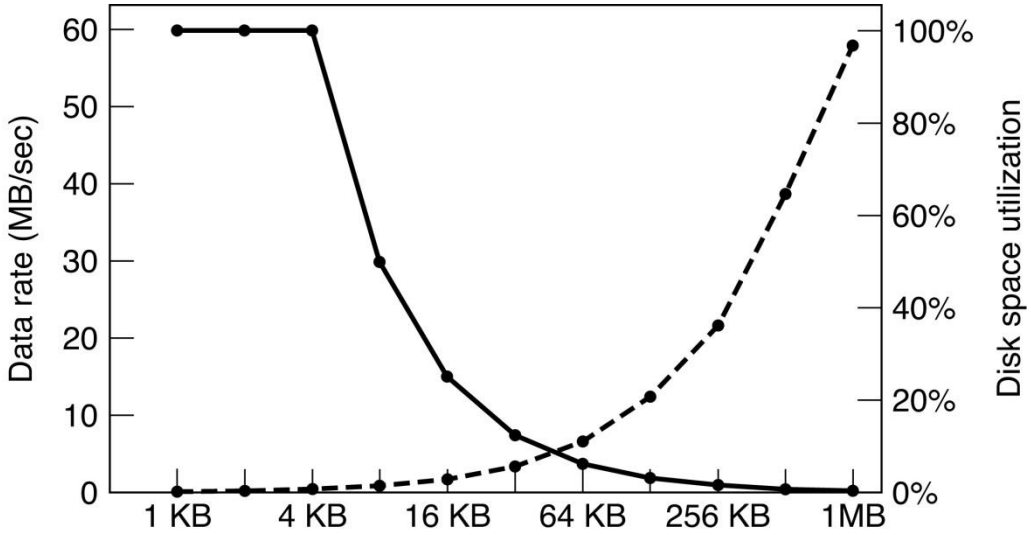
File-system beheer & optimalisatie

- Block grootte
  - Wat is de optimale block grootte?
    - Sector?
    - Track?
    - Cylinder?
    - Page size?
  - Te groot: elke gestarte block is niet meer bruikbaar
    - Kleine files zorgen voor veel verlies
  - Te klein: veel performantieverlies door meer seeks op HDD en dus overhead door hoge seek time en rotatiedelay

erasmus HOGESCHOOL BRUSSEL

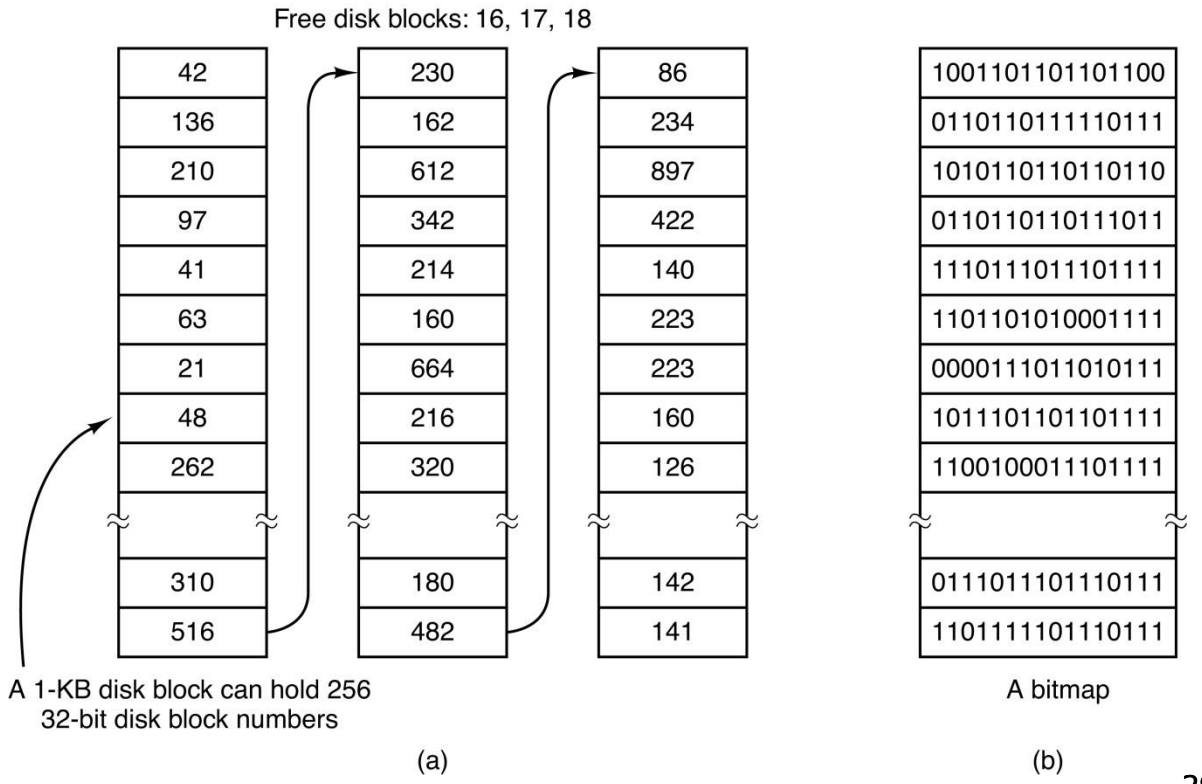
File-system beheer & optimalisatie

- Onderzoek naar grootte van files in praktijksituaties
  - 59% van alle files is 4KB of kleiner
  - 93% van alle blocks wordt gebruikt door de 10% grootste files
  - Onderstaande grafiek met uitsluitend 4KB files
  - Vaste lijn: disk space utilization
  - Stippellijn: data rate
- Historisch: 1-4KB blocks, met grote schijven wordt 64KB interessant



File-system beheer & optimalisatie

- Bijhouden vrije blocks
  - 2 methodes: linked list (a) en bitmap (b)
  - Linked list data wordt in vrije blocks bijgehouden (gratis)



File-system beheer & optimalisatie

- Afhankelijk van het gebruik van de HDD elk zijn voordelen (bij het zoeken naar vrije ruimte)
  - Sterke fragmentatie: bitmap is efficiënter
  - Weinig fragmentatie: LL is efficiënter
  - Ook mogelijkheid om lijst bij te houden van aaneensluitende stukken vrije blocks
  - Moeilijk voorspelbaar wat de beste methode is

erasmus HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- Disk quota's
  - Limieten om gebruik storage door gebruikers
  - Elke file heeft een owner (attribuut)
  - De file-grootte van elke file waarvan de user owner is, wordt bijgeteld op een counter
  - Kan gebruikt worden om quota's te implementeren
  - Soft limit: waarschuwing
  - Hard limit: geen write meer toegestaan

erasmus
HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- File-system backups
  - Data is meest kritieke element in IT-infrastructuur
  - Hardware element dat data bewaart (HDD) is meest crashgevoelig
  - Nood aan backups
    - Recover van disaster (crash,…)
    - Recover van userfout
  - Alles backuppen?
    - OS libraries: kan overgekopieerd worden van install media
    - Temp files: niet nodig
    - POSIX: character en block files: geen echte data en zal fouten genereren
    - Hibernation en pagefile

erasmus HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- Full, incremental of differential backups
  - De archive bit wordt gebruikt
    - » Elk bestand zijn eigen archive bit (attribuut)
    - » Wordt op 1 gezet bij het aanmaken of wijzigen van het bestand
    - » Wordt terug op 0 gezet afhankelijk van het type backup

erasmus HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- **Full**:
  - » negeert de archive bit (A=1 en A=0) bij de selectie van wat te backuppen
  - » reset hem na de backup (A wordt 0)
- **Incremental**:
  - » zal enkel backuppen wat gewijzigd of nieuw is (A = 1)
  - » reset hem na de backup (A wordt 0)
- **Differential**:
  - » zal enkel backuppen wat gewijzigd of nieuw is (A = 1)
  - » wijzigt de archive bit niet na de backup

erasmus
HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

Backup type	Welke files te backuppen? Leest de A bit?	Gaat de A flag resetten?
Full	Nee (negeert)	Ja
Incremental	Ja	Ja
Differential	Ja	Nee

erasmus HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- Compressie
  - Beperkt de grootte hoeveelheid storage vereist
  - Riskant: bij 1 fout alle data mogelijk onbruikbaar
- Backups van live-systemen
  - Offline halen niet altijd mogelijk
  - Snapshot maken van systemstate en alle mogelijke wijzigingen daarna eerst kopiëren en wijzigen ipv dadelijk ter plaatse wijzigen
- Bewaren van backups
  - Disaster op site waar de backups zelf ook mee geraakt worden zoals brand of diefstal
  - Nood aan bewaren van backups off-site

erasmus
HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- Physical dumps
  - Backup van de blocks
  - Lege blocks kunnen overgeslagen worden, maar moeten wel een referentie krijgen om alles terug op zijn plaats te krijgen bij een restore
  - Bad blocks gekend door OS worden tot een bad-block file omgezet zodat deze niet meer wordt gebruikt: backup program moet dit kunnen lezen
  - Onmogelijkheid om files/directories over te slaan en incremental/differential backups uit te voeren
  - Makkelijk en snel

- Logical dumps
  - Backup van de files en directories
  - Makkelijker om specifiek element te restoren
  - Lijst wordt eerst gemaakt van alle te backuppen files via de modified/archive attribuut-bit

erasmus
HOGESCHOOL BRUSSEL

File-system beheer & optimalisatie

- Defragmentatie
  - Initieel alle files aaneensluitend op disk
  - Bij het verwijderen van files ontstaan gaten
  - Worden opgevuld door stukken file
  - Ook groeiende files zonder ruimte geraken gefragmenteerd
  - Defragmentatie: files terug aansluitend maken en trachten alle vrije ruimte in 1 groot stuk op de schijf te krijgen
  - Werkt beter met meer vrije ruimte op het eind van de disk
  - Sommige files kunnen niet gedefragmenteerd worden, vb paging file, journaling log,…
  - Nooit op een SSD!

erasmus HOGESCHOOL BRUSSEL

Overzicht filesystemen

Windows

- FAT:
  - Compatibel met alle OS'en
  - Gebruikt File Allocation Table
  - FAT12: diskettes
  - FAT16 (ook gewoon FAT): max 2GB partitiegrootte
  - FAT32:
    - » max 4GB file grootte
    - » In Windows max 32GB partitiegrootte, maar kan gebruik maken van groter (tot 2TB)
- ExFAT:
  - Opvolger en vervanger van FAT32
  - Geoptimaliseerd voor sdcards en usb-drives
  - Geen (realistische) groottebeperkingen
  - Grote compabiliteit: Windows, MacOS en Linux

erasmus HOGESCHOOL BRUSSEL

Overzicht filesystemen

Windows

- NTFS:
  - veel extra features (compressie, file-en folder level security, encryptie)
  - Journaling
  - Geen (realistische) groottebeperkingen
  - Gebruikt I-nodes
- ReFS:
  - Vanaf Windows Server 2012
  - Bedoeld om hogere betrouwbaarheid te garanderen door copy-on-write: alle metadata die geschreven moet worden, wordt gekopieerd naar een nieuwe file i.p.v. een bestaande te overschrijven. Oude metadata wordt gelinkt aan de nieuwe.

erasmus
HOGESCHOOL BRUSSEL

Overzicht filesystemen

MacOS

- HFS+:
  - geen (realistische) groottebeperkingen
  - Ondersteuning van hard en soft links
  - Journaling
  - Vorm van preventief fragmentatie vermijden

- APFS
  - geen (realistische) groottebeperkingen
  - Sinds MacOS High Sierra de standaard (automatische installatie)
  - Geoptimaliseerd voor flash-based storage (performantie, betrouwbaarheid)
  - Native encryptie, cloning, hoger maximum aantal files,…

erasmus HOGESCHOOL BRUSSEL

Overzicht filesystemen

Linux

- EXT2, 3 en 4:
  - EXT3 is extensie van EXT2 met journaling
  - EXT4 voegt extra attributen toe
  - Zijn case-sensitive
  - Hard-en softlinks
- ReiserFS
  - Geoptimaliseerd voor vele kleine files op te slaan

erasmus
HOGESCHOOL BRUSSEL

Overzicht filesystemen

Linux

- ZFS:
  - Zowel een filesysteem als een logical volume manager
    - » Zal zowel de volumes gaan beheren, alsook het filesysteem erop
    - » Betere beveiliging tegen filecorruptie
  - Voordelen:
    - » Goede beveiliging tegen filecorruptie
    - » Scalable
    - » Goede ondersteuning van opslag met hoge capaciteit
    - » Snapshots
    - » Copy-on-write (zie ook ReFS)
    - » Ingebouwde RAID features (RAID-Z)