Operační systémy

Systémy souborů I

Jan Trdlička

České vysoké učení technické v Praze, Fakulta informačních technologií
Katedra počítačových systémů

https://courses.fit.cvut.cz/BI-OSY

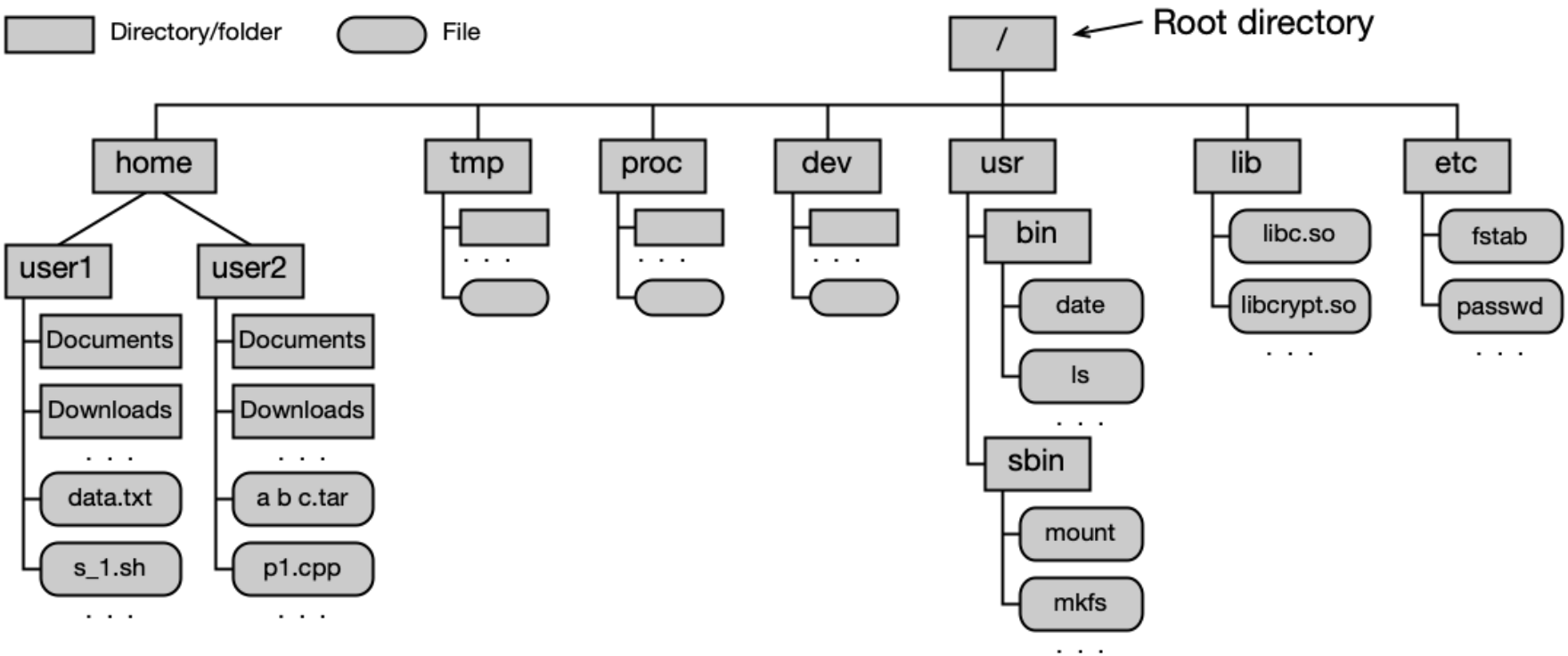
Obsah přednášky

1. FS z pohledu uživatele
2. FS z pohledu administrátora
3. Implementace FS
   - Rozložení dat ve FS
   - Informace o volných datových strukturách
   - Alokace obsahu souboru
     - po souvislých oblastech datových bloků
     - po jednotlivých datových blocích (FAT/i-node)
   - Implementace adresářů
   - Sdílené soubory
   - Příklady: FAT, UFS

FS z pohledu uživatele

- **Uživatel**
  - Data uložená v různých systémech souborů vidí jako jeden (Unix) nebo několik stromů adresářů (MS Windows), které pro něj představují homogenní strukturu.
  - K těmto datům přistupuje prostřednictvím stejných aplikací (GUI), příkazů (CLI) nebo systémových volání/funkcí (API), bez ohledu na to, jestli data jsou uložena na lokálním, pseudo nebo vzdáleném FS.

- **Příklad stromu adresářů v OS unixového typu**



FS z pohledu uživatele

- **Adresáře/Složky**
  - Umožňují hierarchické uspořádání dat do stromové struktury podle požadované logiky.
  - Pozici adresáře/souboru ve stromu definujeme cestou, kterou můžeme vyjádřit dvěma způsoby
    - ★ absolutní cesta: vztažená ke kořeni stromu (kořenový adresář),
    - ★ relativní cesta: vztažená k aktuální pozici ve stromu (pracovní adresář).
  - Jako oddělovač adresářů v cestě se používají v různých OS různé znaky
    - ★ MS Windows: `C:\Users\User1\Documents\data.txt`
    - ★ OS unixového typu: `/home/user1/data.txt`
- **Typické operace nad adresářem**
  - **`Create()`:** vytvoří nový adresář (v Unixu adresář obsahuje podadresáře `.` a `..`).
  - **`Delete()`:** smaže adresář.
  - **`Opendir()`:** do hlavní paměti se z FS načtou potřebné informace o adresáři (atributy, diskové adresy,...).
  - **`Closedir()`:** uvolní se příslušné datové struktury v hlavní paměti, které související s adresářem.
  - **`Readdir()`:** načte se následující položka adresáře.
  - **`Seekdir()`:** nastaví pozici následující položky pro `Readdir()`.
  - **`Rename()`:** přejmenuje adresář.
  - **`Link()`:** vytvoří link na adresář.
  - **`Unlink()`:** smaže link na adresář.

FS z pohledu uživatele

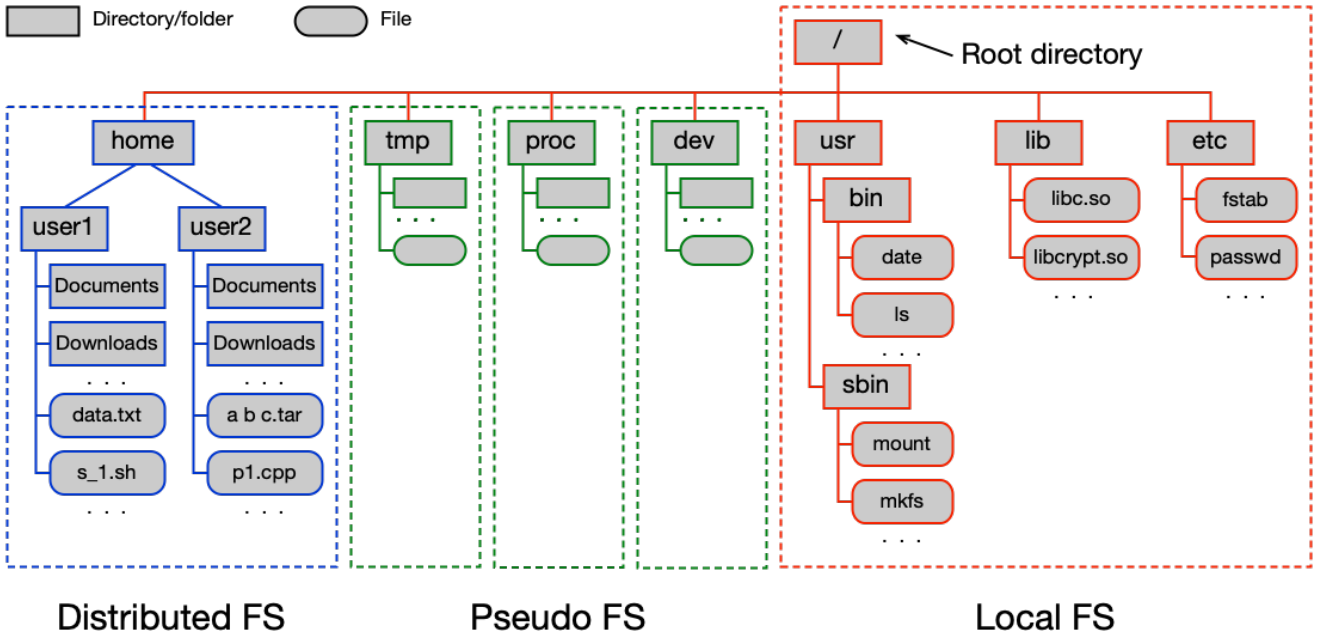
- **Soubor**
  - Slouží k uložení informace/dat ve FS a k jejímu pozdějšímu použití.
  - Soubor je ve FS reprezentován: jménem, atributy a obsahem.
  - **Jméno souboru**
    - ★ Jméno souboru společně s cestou slouží k určení pozice ve stromě adresářů.
    - ★ V různých FS můžou být různé požadavky na jméno (typ kódování, rozlišování malých/velkých písmen, používání speciálních znaků, maximílní délka,...).
  - **Atributy souboru**
    - ★ Definují vlastnosti souboru a patří mezi ně následující atributy.
    - ★ Typ: obyčejný soubor, adresář, link,...
    - ★ Vlastníci souboru: uživatel, skupina, ostatní,...
    - ★ Přístupová práva: pro čtení, modifikaci a spuštění, setuid-bit, ACL práva,...
    - ★ Různé časy: čas přístupu, modifika,...
  - **Obsah souboru**
    - ★ Představuje samotnou informaci/data, která jsou uložena v datových blocích ve FS.
    - ★ Většina OS (MS Windows, Unix,...) vidí obsah souboru pouze jako pole bytů a jeho interpretace je na jednotlivých procesech (vyjímkou jsou spustitené soubory).

FS z pohledu uživatele

- **Typické operace nad souborem**
  - **`Create()`:** vytvoří nový soubor a nastaví některé z atributů.
  - **`Delete()`:** smaže soubor.
  - **`Open()`:** do hlavní paměti se z FS načtou potřebné informace o souboru (atributy, diskové adresy,...).
  - **`Close()`:** uvolní se příslušné datové struktury v hlavní paměti, které související se souborem.
  - **`Read()`:** načte příslušný počet bytů od aktuální pozice.
  - **`Write()`:** zapíše příslušný počet bytů od aktuální pozice.
  - **`Seek()`:** nastaví aktuální pozici na novou hodnotu.
  - Funkce pro načtení atributů: `stat()`, `access()`,...
  - Funkce pro nastavení atributů: `chmod()`, `chown()`,...
  - **`Rename()`:** přejmenuje soubor.

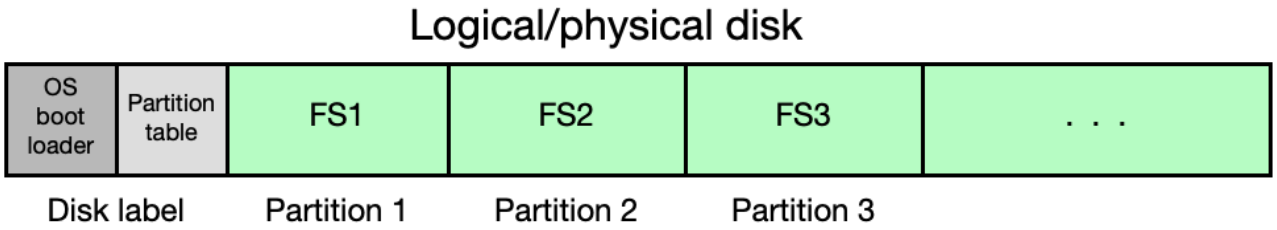
FS z pohledu administrátora

- **Strom adresářů**
  - Celý strom adresářů může reprezentovat pouze jeden FS nebo několik FS propojených do hromady.
- **Příklad stromu adresářů v OS unixového typu**
  - Typicky obsahuje několik FS
    - ★ Do kořenového adresáře je obvykle připojen lokální FS.
    - ★ Do podadresářů `/proc`, `/tmp`, `/dev` jsou připojeny příslušné pseudo FS (existují pouze v hlavní paměti, nikoliv na disku).
    - ★ Do podadresáře `/home` může být připojen distribuovaný FS (data jsou uložená na vzdáleném serveru).



FS z pohledu administrátora

- **Rozložení dat na disku (disk layout)**



  - Logický/fyzický disk je typicky rozdělen na několik částí, které obsahují informace/data s různým významem pro OS.

1. Label disku (disk label)
   - ★ Nachází se na začátku disku a obsahuje informace o rozdělení disku na jednotlivé oblasti (Partition table).
   - ★ Může také obsahovat zavaděč OS (OS boot loader).
   - ★ V praxi existuje několik různých formátů: MBR (Master Boot Record), EFI GPT (Extensible Firmware Interface GUID Partition Table),...
2. Jednotlivé diskové oblasti (partitions/slices)
   - ★ Každá disková oblast obsahuje jeden FS.

- Rozdělení disku na jednotlivé diskové oblasti, instalace zavaděče OS, vytvoření jednotlivých FS v příslušných oblastech se typicky provádí při instalaci systému a odpovídá za to administrátor systému. Pozdější změna rozložení dat na disku může být komplikovaná.

FS z pohledu administrátora

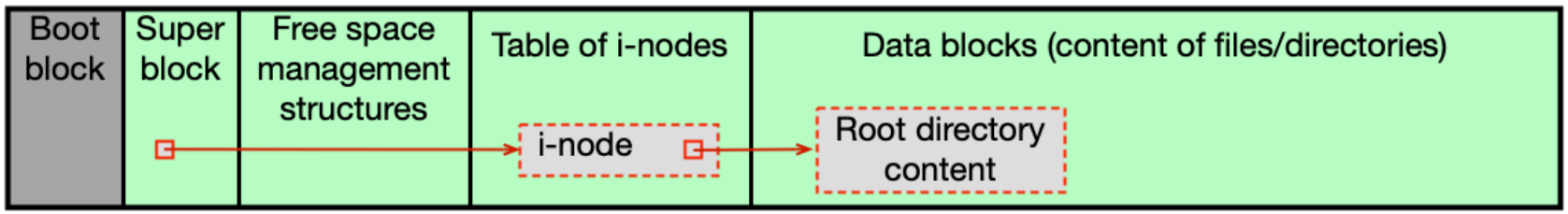
- Během instalace se také definuje, do kterých adresářů (přípojných bodů) ve stromu adresářů se budou jednotlivé FS připojovat. Toto lze relativně jednoduše upravit i později po instalaci systému.
- **Typické operace nad diskem a FS**
  - Rozdělení disku: např. příkazy `fdisk` (Linux), `format` (Solaris),...
  - Vytvoření FS: varianty unixových příkazů `mkfs`, `mkfs.ext4`, `mkfs.vfat`,...
  - Zvětšení FS: příkaz `growfs` (Solaris ufs),...
  - Kontrola FS: příkaz `fsck`,...
  - Připojení FS do stromu adresářů: příkaz `mount`
  - Odpojení FS: příkaz `umount`
  - Vytvoření zálohy FS: příkaz `dump` (Linux ext2/3/4), `ufsdump` (Solaris ufs),...
  - Obnova dat ze zálohy: příkaz `restore` (Linux ext2/3/4, Solaris ufs),...

Implementace FS

- **Rozložení dat ve FS (FS layout)**
  - V oblasti disku, ve které je vytvořen FS, se obvykle nachází následující typy informací/dat.

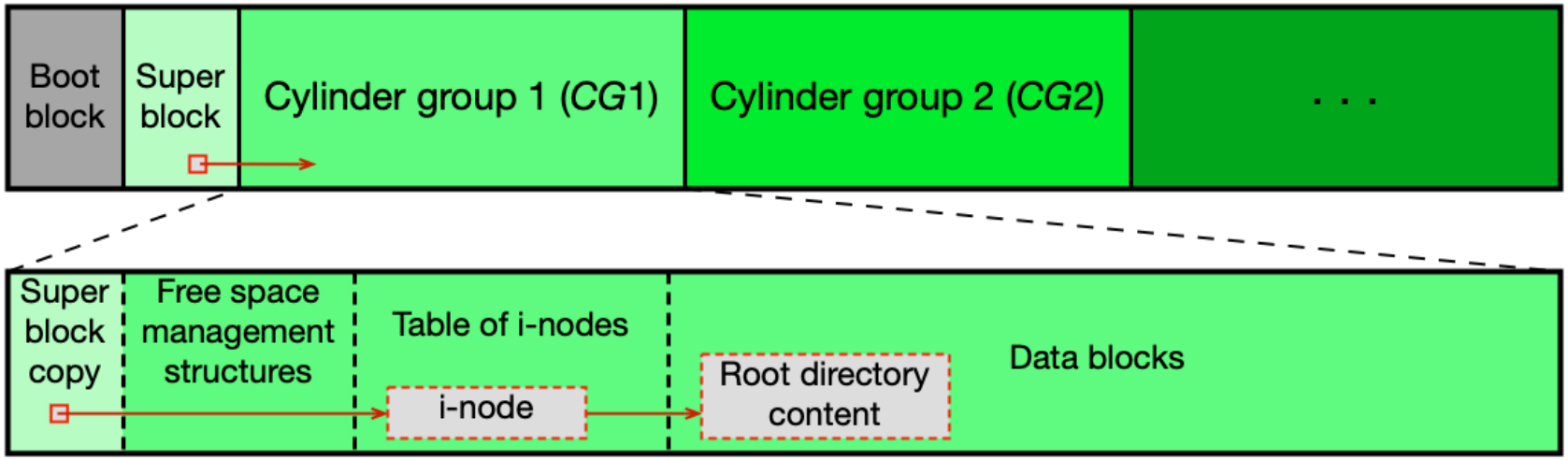
1. Kód sloužící k zavedení OS (Boot block).
2. Informace popisující konfiguraci FS (Super block)
   - ★ Typ FS, Velikost datových bloků.
   - ★ Informace o celkové velikosti a aktuální obsazenosti FS (např. počet i-nodů, datových bloků,...).
   - ★ Informace o diskových adresách důležitých struktur FS.
3. Datové struktury pro správu volného prostoru (datových bloků, i-nodů,...).
4. Datové struktury pro uložení atributů souborů (Tabulka i-nodů,...).
5. Datové bloky, do kterých se ukládá obsah souborů a adresářů.

- **Příklad rozložení dat v jednoduchém FS**



Implementace FS

- **Příklad rozložení dat v UFS (Unix File System)**



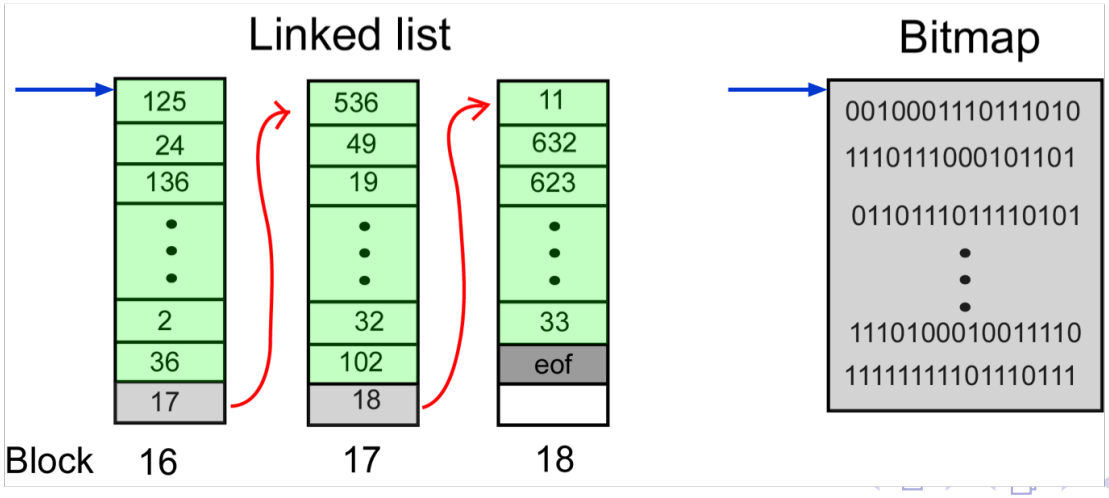
  - UFS je reálný FS používaný např. v Solarisu nebo BSD.
  - Z důvodu výkonu a zabezpečení dat proti ztrátě je diskový prostor UFS rozdělen do několik stejně velkých oblastí CG_i (Cylinder groups), které jsou reprezentovány souvislou množinou cylindrů na disku.
  - Soubor/adresář je vždy alokován v rámci konkrétní $CG_i \Rightarrow$ lepší výkon (hlavičky HDD se pohybují pouze v rámci CG_i).
  - Pokud dojde k poškození začátku disku (např. administrátor přepíše omylem Super blok a několik prvních sektorů z CG_1), pak se ztratí data z CG_1, ale data z ostatních CG_i se podaří většinou zachránit pokud víme s jakými parametry byl UFS vytvořen.

Implementace FS

- **Sector**
  - Nejmenší adresovatelná jednotka datového úložiště (4 KB/512 B).
- **Datový blok**
  - Velikost sektoru je obvykle příliš malá z hlediska FS a navíc je fixní pro dané datové úložiště.
  - Ve FS se proto používá alokace dat po větších logických jednotkách, které se nazývají datové bloky ("clusters" v MS Windows).
  - Velikost datového bloku může definovat administrátor při vytvoření FS a je závislá na velikosti samotného FS a na očekávané velikosti ukládaných dat.
- **Příklad**
  - UFS (Unix file system) v Solarisu má pouze 8KB datové bloky.
  - Unixový VXFS (Veritas FS) má 1, 2, 4, 8 KB datové bloky.
  - FAT32 v MS Windows má 4KB,...,32KB datové bloky.
  - NTFS v MS Windows má 4KB až 2 MB datové bloky.

Implementace FS

- **Správa volných datových struktur**
  - Podobně jako při správě paměti i zde lze použít pro správu volných datových struktur buď bitovou mapu nebo zřetězený seznam.
  - Zřetězený seznam
    - ★ Je uložen přímo ve volných datových blocích FS.
    - ★ Po připojení FS je obvykle nahrána do hlavní paměti pouze část tohoto seznamu.
  - Bitová mapa
    - ★ Většinou zabírá méně místa než zřetězený seznam.
    - ★ Pouze v případě téměř zaplněného FS je zřetězený seznam výhodnější z hlediska velikosti.



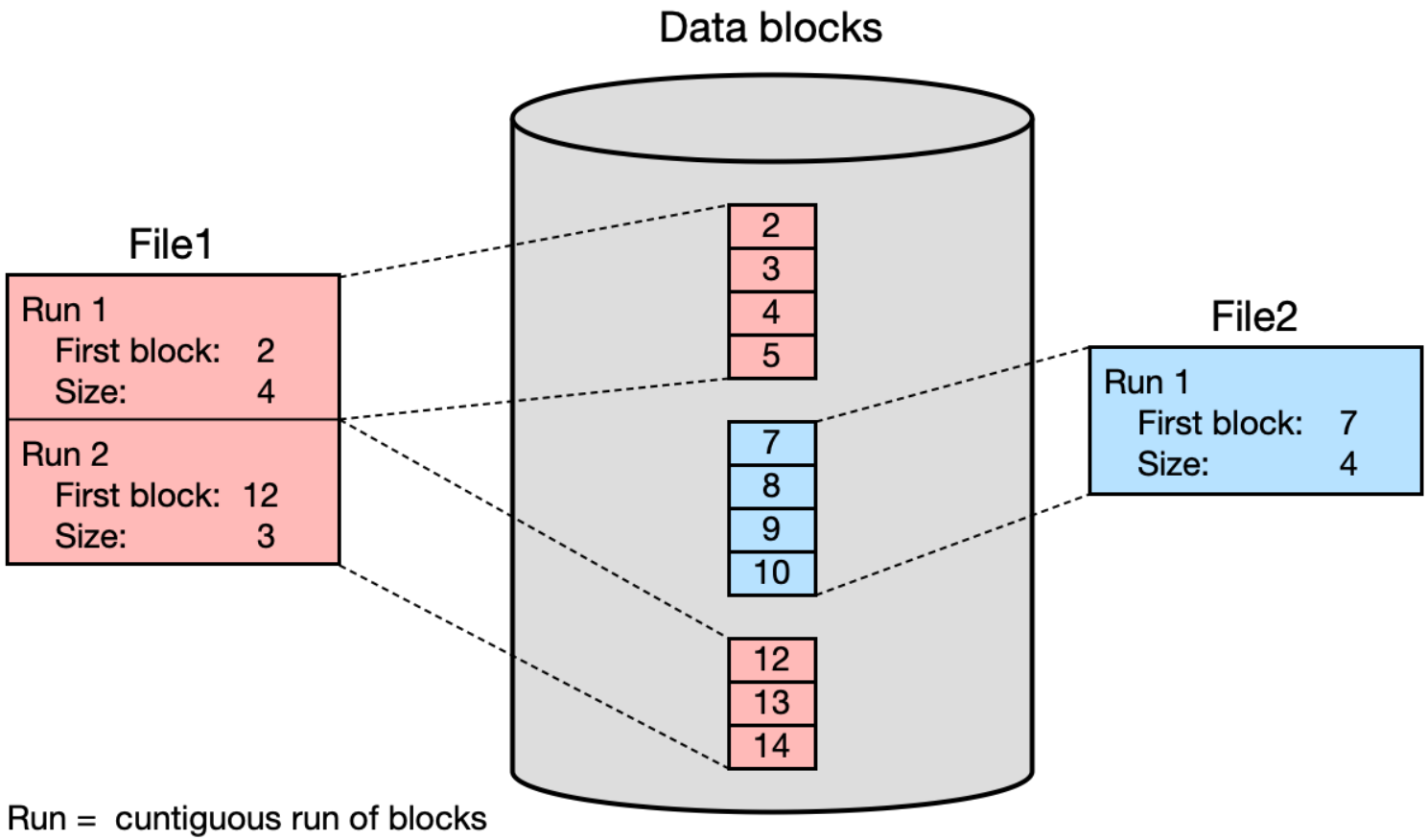
Implementace FS

- **Obsah souborů/adresářů je uložen v datových blocích**
  - Z hlediska výkonu FS je klíčový způsob alokace datových bloků.
  - Rozlišujeme dva základní přístupy
    1. alokace souvislé oblasti datových bloků (contiguous run of blocks),
    2. alokace po jednotlivých datových blocích.

1. **Alokace souboru pomocí souvislých oblastí datových bloků**
   - V ideálním případě je obsah souboru uložen v jedné souvislé oblasti bloků. Protože se v čase velikost souboru většinou mění, tak častěji je jeho obsah uložen v několika souvislých oblastech bloků.
   - **Výhody**
     - ★ Pro každý soubor si FS musí pamatovat malý počet informací (např. adresu prvního bloku a počet bloků).
     - ★ Výborný výkon při sekvenčním přístupu a u velkých souborů.
   - **Nevýhody**
     - ★ Problém s fragmentací FS
       ⇒ složitější alokace souvislé oblasti,
       ⇒ je nutné pravidelně defragmentovat.
   - **Příklady**
     - ★ VXFS (Veritas File System),
     - ★ NTFS (New Technology File System),...

Implementace FS

- **Příklad: Alokace obsahu souboru pomocí souvislý oblastí**
  - Obsah souboru File1 je uložený ve dvou souvislých oblastech, které se skládají z bloků 2, . . . , 5 a 12, . . . , 13.
  - Obsah souboru File2 je uložený v jedné souvislé oblasti, která je složena z bloků 7, . . . , 10



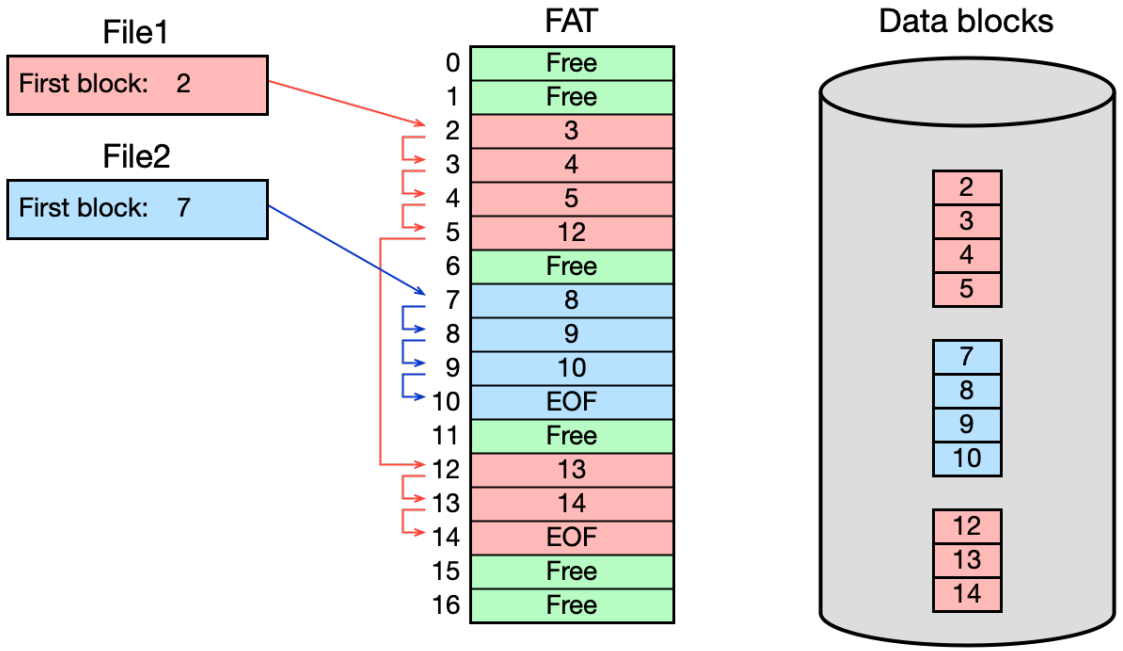
Implementace FS

2 Alokace souboru po jednotlivých datových blocích

- Obsah souboru je alokován po jednotlivých blocích
  $\Rightarrow$ FS si musí pamatovat adresy všech bloků, kde je uložený obsah souboru.
- Pro uložení adres bloků se používají následující struktury
  a. FAT (File Allocation Table),
  b. i-node (index node).
- Přestože se obsah souboru alokuje po jednotlivých blocích, ovladač FS se snaží uložit obsah souboru v rámci jednoho nebo několika sousedních cylindrů tak, aby se minimalizoval čas vystavení hlaviček u HDD.
- **Výhody**
  - ★ Není problém s fragmentací.
  - ★ Dobrý výkon při náhodném přístupu a u malých souborů.
- **Nevýhody**
  - ★ Pro každý soubor si FS musí pamatovat velký počet informací.
  - ★ Horší výkon při sekvenčním přístupu.
- **Příklady**
  - ★ FAT32 (File Allocation Table),
  - ★ UFS (Unix File System),
  - ★ EXT2/3/4 (Extended File System),...

Implementace FS

- **FAT (File Allocation Table)**
  - Umožňuje alokaci obsahu souboru po jednotlivých datových blocích.
  - Tabulka obsahuje tolik řádek kolik je datových bloků ve FS.
  - Pro každý soubor si musíme pamatovat pouze adresu prvního bloku, adresy dalších bloků jsou uloženy ve FAT formou zřetězení.
  - Každý řádek *i* tabulky obsahuje právě jednu z následujících hodnot
    - ★ Free: datový blok *i* je volný,
    - ★ Adresa: adresa následujícího bloku za blokem *i*, kde pokračuje obsah souboru,
    - ★ EOF: konec zřetězení (blok *i* je posledním blokem souboru).

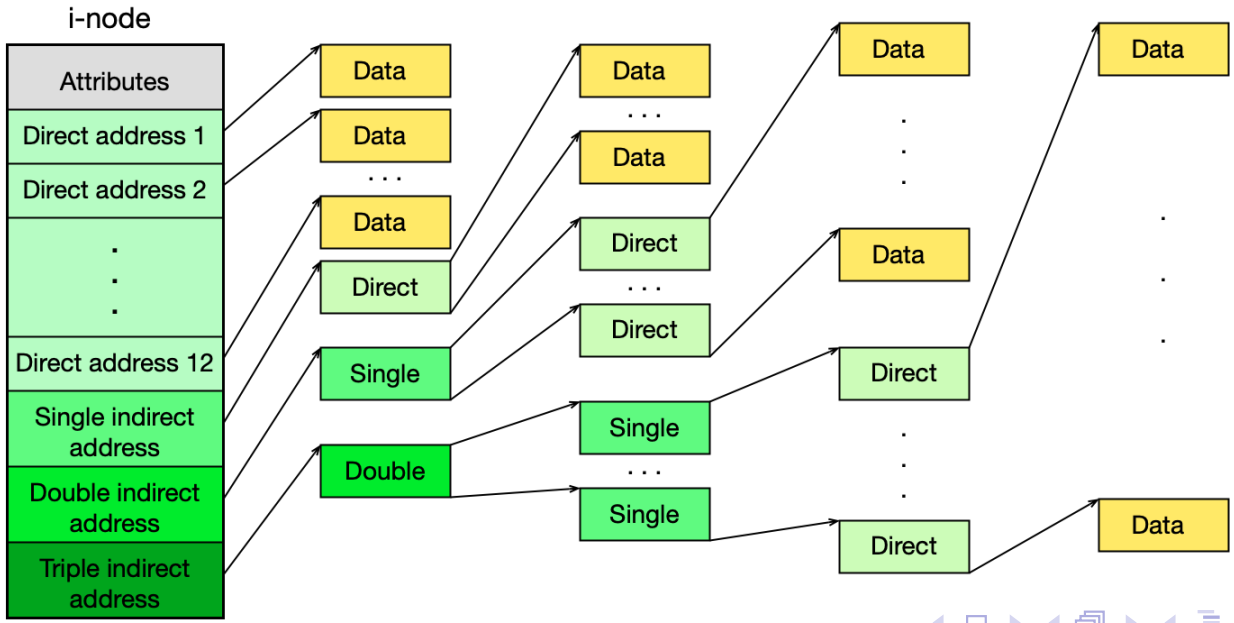


Implementace FS

- **Vlastnosti FAT**
  - Informace o volných datových blocích se dají vyčíst přímo z FAT.
  - Při připojení systému s FAT se do hlavní paměti načte celá/část FAT $\Rightarrow$ urychlí se přístup k obsahu souboru.
  - Pro velké disky vzniká problém s velikostí FAT, protože velikost datových bloků se v čase příliš nenavyšuje.
- **Příklad**
  - Předpokládejme následující parametry
    - ★ Velikost datových bloků je 1KB.
    - ★ Velikost adresy je 32 bitů.
  - Z těchto parametrů vyplývají následující vlastnosti.
    - ★ FAT může mít až 2^{32} řádek.
    - ★ Jedna řádka zabírá 32 bitů = 4 B.
    - ★ Velikost FAT může být až $2^{32} \times 4B = 2^{34} = 16GB$.
    - ★ Maximální velikost FS může být teoreticky až $2^{32} \times 1KB = 2^{42}B = 4TB$.
    - ★ Maximální velikost souboru může být teoreticky až $2^{32} \times 1KB = 2^{42} = 4TB$.
    - ★ POZOR: maximální velikosti a maximální počty mohou být omezeny velikostí ostatních struktur ve FS nebo OS.

Implementace FS

- **I-node (Index node)**
  - S každým souborem/adresářem je spojen příslušný i-node.
  - I-node je datová struktura, která má fixní velikost a ve které jsou uloženy atributy souboru/adresáře a adresy datových bloků, kde je uložen jeho obsah.
  - **Z důvodu adresace různě velkých souborů jsou zde uloženy tři typy adres**
    - ★ 12 přímých adres: ukazující přímo na datové bloky, kde je obsah souboru,
    - ★ 1 nepřímá adresa první úrovně: ukazuje na blok, ve kterém jsou přímé adresy,
    - ★ 1 nepřímá adresa druhé úrovně: ukazuje na blok, ve kterém jsou nepřímé adresy první úrovně,
    - ★ 1 nepřímá adresa třetí úrovně: ukazuje na blok, ve kterém jsou nepřímé adresy druhé úrovně.



Implementace FS

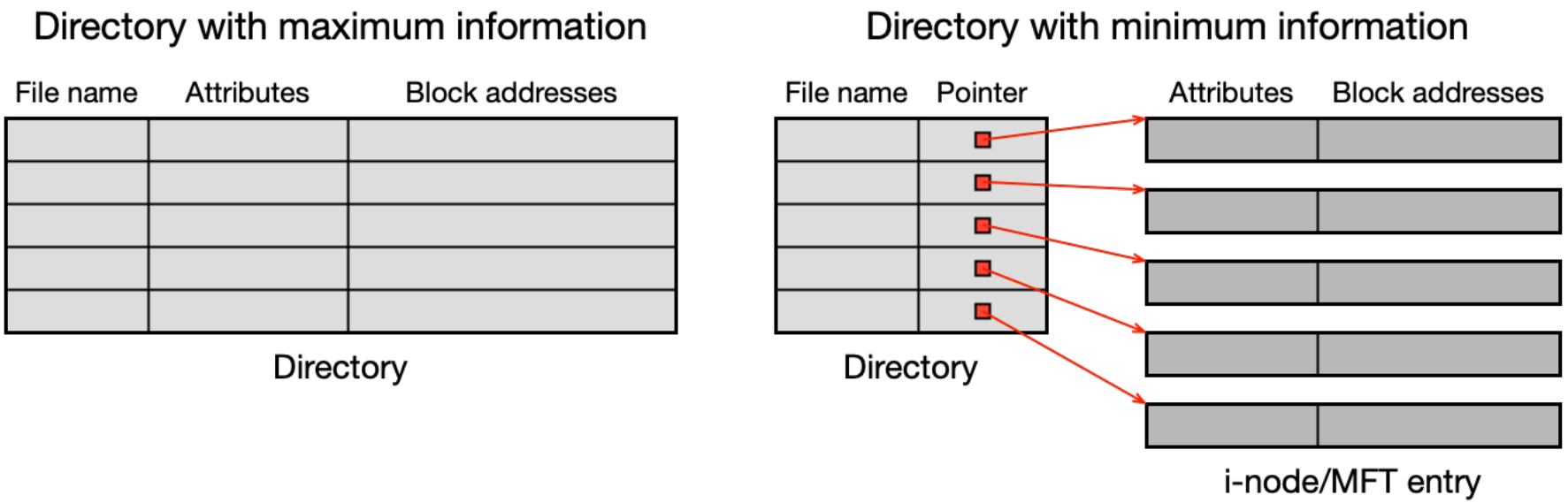
- **Vlastnosti i-nodů**
  - Při otevření souboru/adresáře se do hlavní paměti načte pouze jeho i-node a teprve při čtení/zápisu se načítají jednotlivé bloky s daty/adresami.
  - Při zápisu do souboru se postupně využívají přímé adresy, nepřímé adresy první, druhé a nakonec třetí úrovně v závislosti na velikosti souboru.
  - U velkých souborů a náhodném přístupu je pomalejší přístup k datům při první přístupu. Při následujících přístupech se již využívá skrytá paměť.
  - Horší využití prostoru FS, protože část datových bloků se používá na metadata (bloky s adresami).
  - Samotná velikost i-nodu není závislá na velikosti FS nebo velikosti souboru.
  - Počet i-nodů se implicitně odvozuje od kapacity FS. V řadě FS je počet i-nodů statický (po vytvoření FS nelze počet navýšit), např. UFS, EXT2/3/4, VxFS,...
- **Příklad**
  - Předpokládejme následující parametry
    - ★ Velikost datových bloků je 4KB.
    - ★ Velikost adresy je 32 bitů.
  - Z těchto parametrů vyplývají následující vlastnosti.
    - ★ Počet adres v bloku je $4KB/32bit = 2^{10}$.
    - ★ Maximální velikost souboru může být teoreticky

$$(12 + 2^{10} + 2^{10} \times 2^{10} + 2^{10} \times 2^{10} \times 2^{10}) \times 4KB \approx 2^{30} \times 4KB = 4TB$$

    - ★ POZOR: maximální velikosti a maximální počty mohou být omezeny velikostí ostatních struktur ve FS nebo OS.

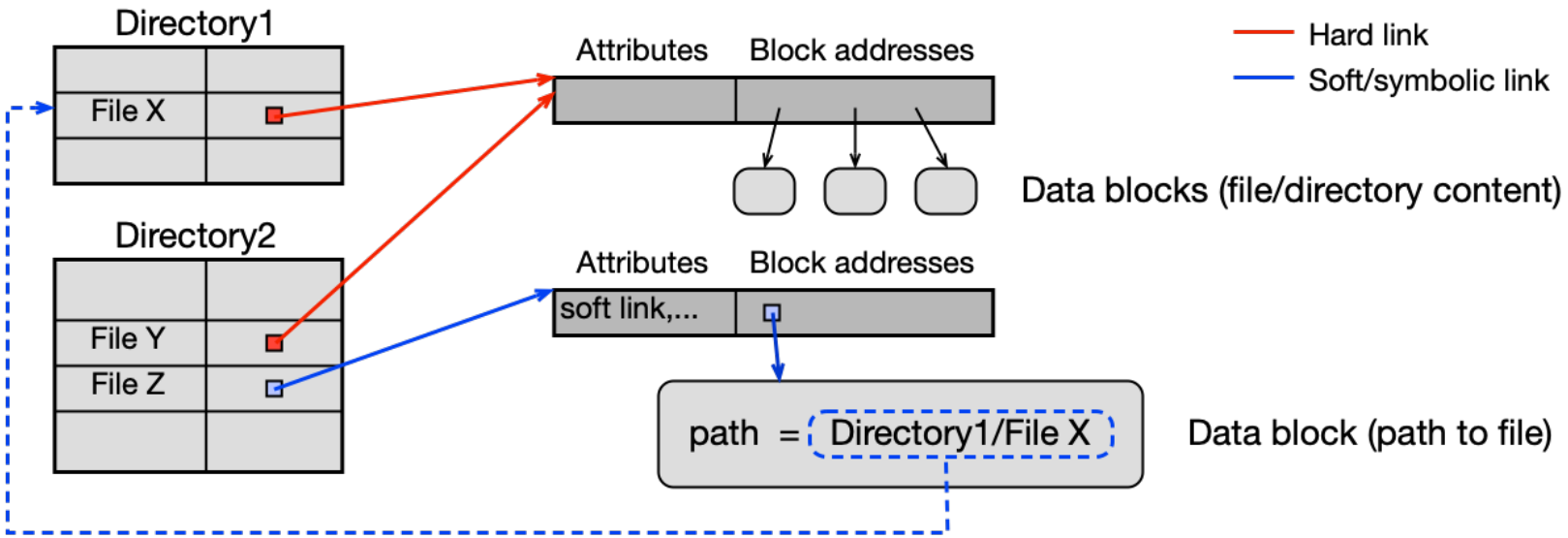
Implementace FS

- **Adresáře**
  - Obsah adresářů je podobně jako obsah souborů uložen v jednom nebo několika datových blocích.
  - **Z hlediska implementace existují dva přístupy.**
    - ★ Adresář obsahuje většinu informací o souborech a podadresářích (jméno, atributy a adresy bloků s obsahem souboru), např. FAT32.
    - ★ Adresář obsahuje minimum informací (jméno a odkaz do "speciální" datové struktury (např. i-node v UFS nebo položka Master File Table v NTFS), ve které jsou uloženy ostatní informace).



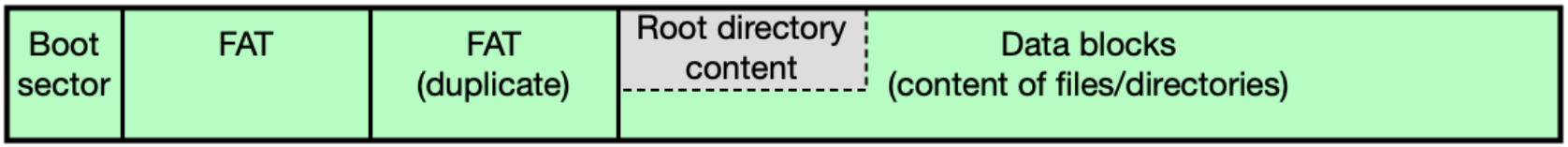
Implementace FS

- **Sdílené soubory**
  - Občas je vhodné, aby ten samý soubor/adresář byl současně viditelný v různých adresářích (popř. i pod různými jmény).
  - Většina současných FS toto umožňuje a k dispozici jsou dvě implementace.
    - ★ Soft link: odkaz na existující soubor/adresář je implementován pomocí cesty k souboru.
    - ★ Hard link: implementován pouze u minimalistických adresářů přímo pomocí odkazu na "speciální datovou strukturu" (i-node/MFT položku).



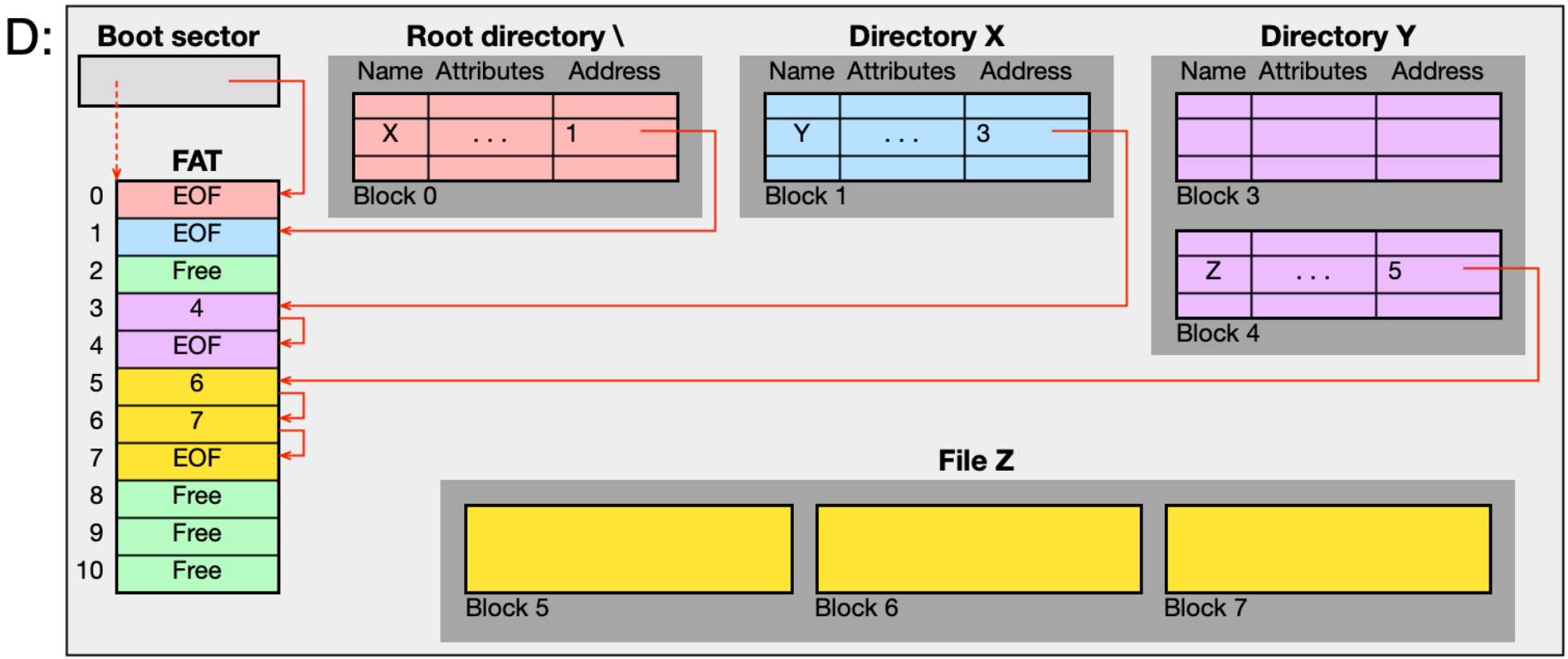
Příklad: FAT32/exFAT

- **Popis**
  - Existuje několik různých implementací systém souborů FAT podle velikosti adresy
    - ★ FAT12 (12 bitová verze),
    - ★ FAT16 (16 bitová verze),
    - ★ FAT32 (32 bitová verze používající 28 bitů),
    - ★ exFAT (64 bitová verze).
  - Obsah adresáře je implementován jako tabulka a je uložen v jednom nebo několika datových blocích.
  - Záznam o souboru/podadresáři je obvykle uložen v jedné řádce tabulky, která má statickou strukturu a obsahuje
    - ★ jméno souboru,
    - ★ atributy souboru (typ, velikost,...),
    - ★ adresu prvního bloku, kde začíná obsah souboru/podadresáře (informace o dalších blocích, kde pokračuje obsah, jsou ve FAT).
  - Pokud se některé informace (jméno souboru, ACL práva,...) nevejdou do vyhrazeného místa mohou být uloženy ve více řádkách adresáře.
  - Rozložení dat na disku u FAT32
    - ★ exFAT obsahuje navíc bitovou mapu s informací o volných datových blocích.



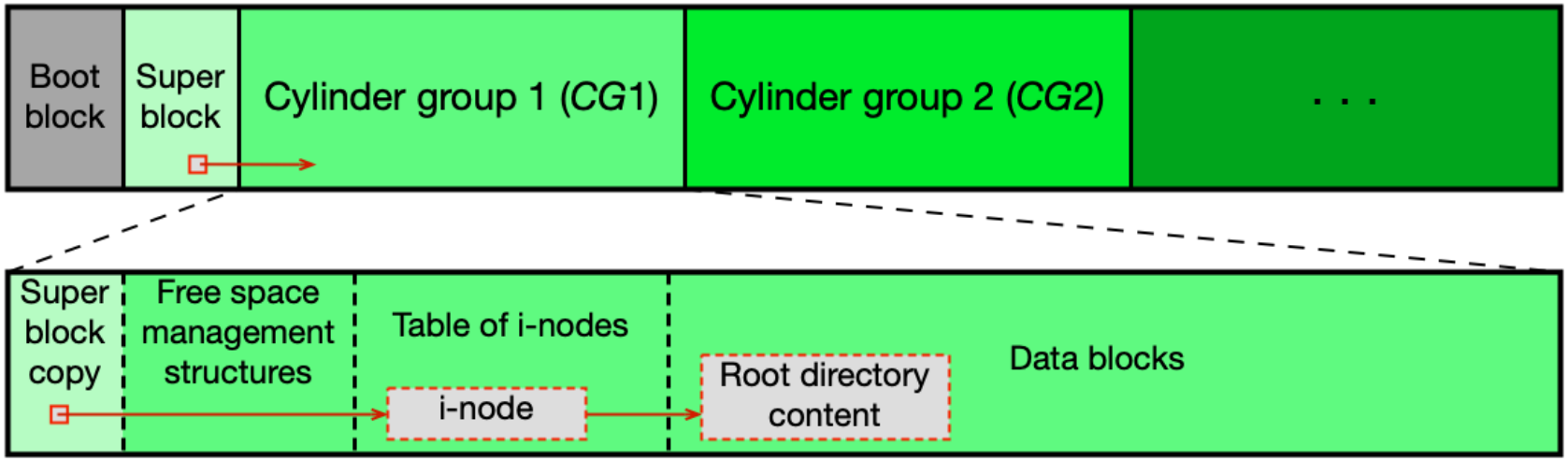
Příklad: FAT32/exFAT

- **Přístup k adresářů/souborů**
  - Boot sektor obsahuje informaci, kde začíná FAT, kopie FAT a obsah kořenového adresáře (Root directory).
  - Při připojení FS se načte do hlavní paměti celé/část FAT.
  - Pokud chceme např. zobrazit obsah souboru `D:\X\Y\Z`, pak musíme načíst z disku příslušné datové bloky.



Příklad: UFS (Unix File System)

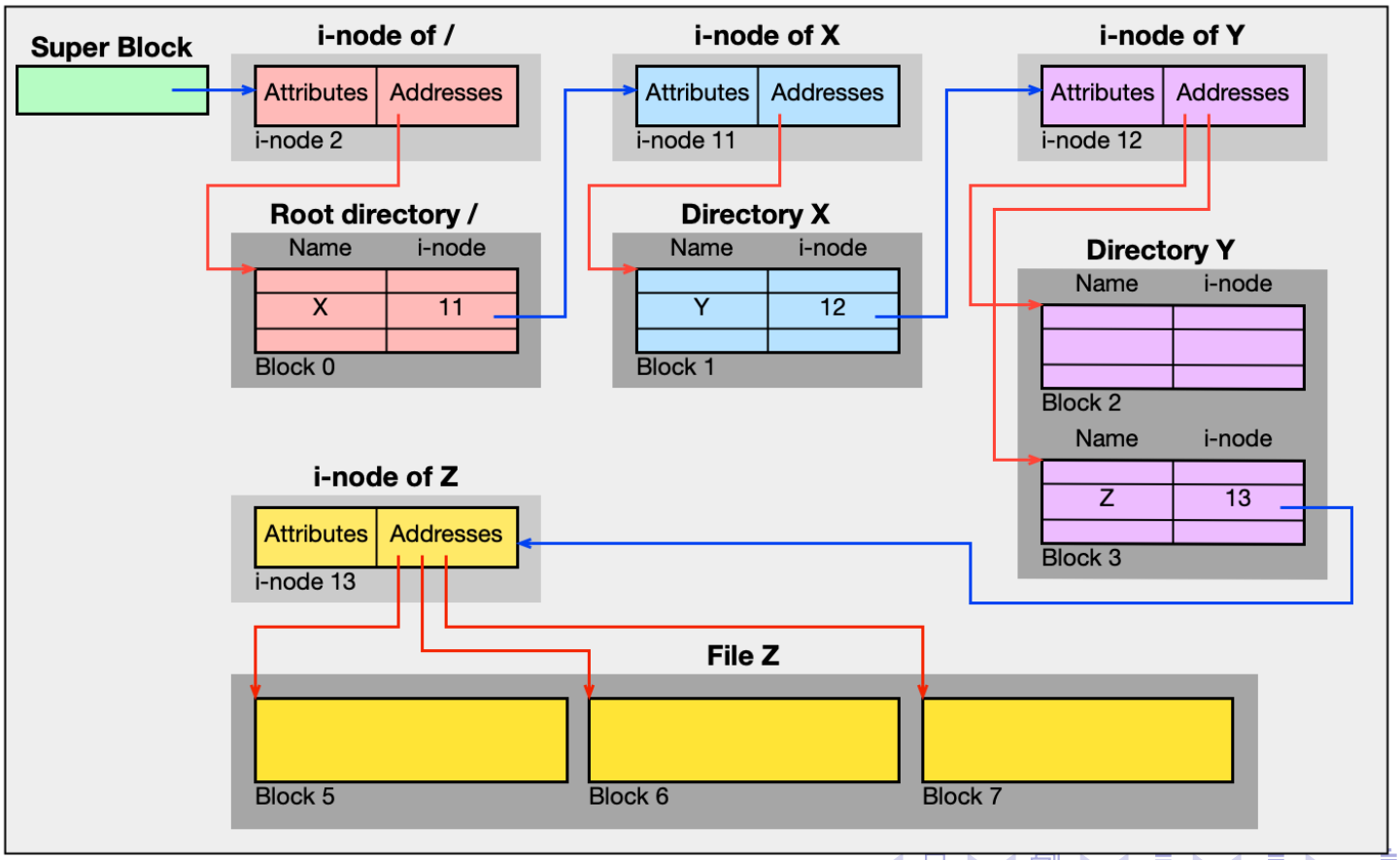
- **Popis**
  - Někdy je také označovaný jako BSD Fast File System (FFS).
  - Můžeme ho najít v různých OS (BSD, Solaris,...) a v různých verzích, ve kterých jsou implementovány nové vlastnosti (např. žurnálování, snapshoty, ACL práva,...).
  - Byly jím inspirovány další FS (HFS+ v MACOS, Ext2/3/4 v Linuxu,...).
  - V OS Solaris je velikost datového bloku 8KB, adresář obsahuje jméno souboru a číslo i-node daného souboru/podadresáře.
  - i-node má velikost 128 B a obsahuje atributy souboru/adresáře (přístupová práva, vlastníka, ...) a 15 32-bitových adres diskových bloků (12 přímých a 3 nepřímé adresy první, druhé a třetí úrovně).
- **Rozložení dat na disku**



Příklad: UFS (Unix File System)

- **Přístup k souborům/adresářům**
  - Super blok obsahuje informaci, kde začínají struktury pro správu volného prostoru, tabulka i-nodů a datové bloky.
  - Po připojení UFS se do paměti načte i-node kořenového adresáře (i-node číslo 2).
  - Pokud chceme např. zobrazit obsah souboru `/X/Y/Z`, pak musíme načíst z disku příslušné i-nody a datové bloky.

Root FS:



Použité zdroje

1. A. S. Tanenbaum, H. Bos: *Modern Operating Systems (4th edition)*, Pearson, 2014.
2. W. Stallings: *Operating Systems: Internals and Design Principles (9th edition)*, Pearson, 2017.
3. A. Silberschatz, P. B. Galvin, G. Gagne: *Operating System Concepts (9th edition)*, Wiley, 2012.
4. R. McDougall, J. Mauro: *Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture (2nd edition)*, Prentice Hall, 2006.