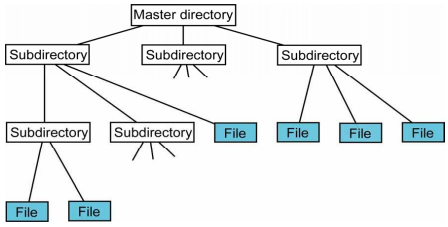
Operační systémy

Přednáška 11: Souborové systémy

Strom adresářů



- **Soubory**
  - umožňují uložení informace na disku..
- **Adresáře/Složky**
  - umožňují hierarchické uspořádání dat na disku.

Soubory

- **Soubor = jméno + atributy + data**

- **Jméno souboru**
  - Délka (8+3, 255,…), kódování (ASCII, UNICODE,…).
- **Atributy souboru**
  - Typ souboru (adresář, obyčejný soubor, link,…).
  - Vlastníci souboru (uživatel, skupina, ostatní,…).
  - Přístupová práva (čtení, zápis, spuštění, setuid, ACL,…).
  - Čas (vytvoření, modifikace, přístupu,…). …
- **Data**
  - Obsah souboru, který je uložen v datových blocích na disku.

- **Přístup k souboru**
  - Pomocí systémových volání: open(), close(), seek(), read(), write(), stat(),...
  - Příkazy OS, aplikace.

Adresáře/Složky

- Umožňují **hierarchické uložení informací** ve stromě adresářů.

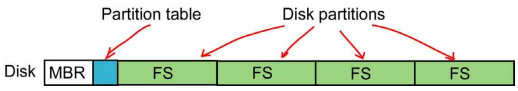
- **Cesta k souboru/adresáři**
  - **Absolutní**
    - Začíná vždy v kořenovém adresáři *root*.
    - Obsahuje posloupnost podadresářů mezi *root* a cílovým souborem.

      /home/rocnik2/skupuna12/Novak

  - **Relativní**
    - Začíná vždy v aktuálním adresáři *current*.
    - Obsahuje posloupnost podadresářů mezi *current* a cílovým souborem.

      current=/home/rocnik2/skupina15
      ../skupina12/Novak
- **Přístup k adresáři**
  - Pomocí systémových volání: opendir(), creat(), getfirts(), getnext(), link()…
  - Příkazy OS, aplikace.

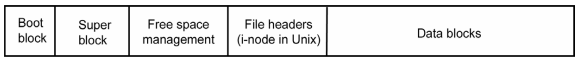
Rozvržení disku



- **Master Boot Record (MBR)**
  - Obsahuje program pro zavedení jádra OS z disku do paměti.

- **Tabulka oblastí (Partition table)**
  - Disk může být rozdělen **na několik diskových oblastí.**
  - Každá oblast může obsahovat svůj systém souborů.

Rozvržení systému souborů (FS)



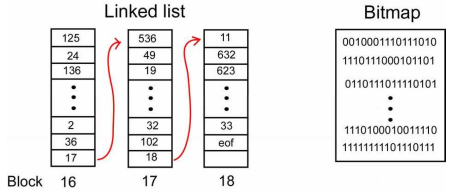
- **Super block** obsahuje klíčové informace o FS:
  - Magické číslo (pro identifikaci typu FS),
  - Velikost datových bloků a jejich počet v daném FS,
  - Velikost “File headers” oblasti,
  - Umístění kořenového adresáře, …
- **Free space management**
  - Informace o volných datových blocích a „file headers“.
- **File headers**
  - Pole struktur (jedna na soubor), obsahujících např. atributy souborů a adresy datových bloků.
- **Datové bloky**
  - Je v nich uložen obsah adresářů a souborů.

Sector vs. datový blok

- **Sector** (fyzická jednotka)
  - Minimální adresovatelná jednotka na disku (typicky 512B).
- **Datový blok** (logická jednotka)
  - Minimální adresovatelná jednotka ve FS (souvislá posloupnost sektorů).
- **Příklad**:
  - UNIX **ufs** má 4kB, 8kB, 16kB, 32kB, 64kB datové bloky.
  - UNIX **vxfs** má 1 kB datové bloky.
  - MS Windows **FAT32** má 4KB,...,32KB datové bloky.
    ntfs má 512B,...,64KB datové bloky.
- **Velké dat. bloky**
  - Vhodné pro FS s velkými soubory a soubory s sekvenčním přístupem.
- **Malé dat. bloky**
  - Vhodné pro FS s malými soubory a soubory s náhodným přístupem.

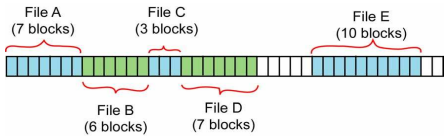
Správa volného prostoru

- **Zřetězený seznam**
  - Je uložen ve volných datových blocích.
  - V hlavní paměti je pouze část tohoto seznamu.
- **Bitová mapa**
  - Většinou zabírá méně místa než zřetězený seznam.
  - Pouze v případě téměř zaplněného FS je zřetězený seznam výhodnější.



Souvislá alokace dat. bloků

- Obsah souboru je uložen na disku v **souvislé posloupnosti dat. bloků.**
- Používá se např. v UNIX Veritas File System (vxfs).

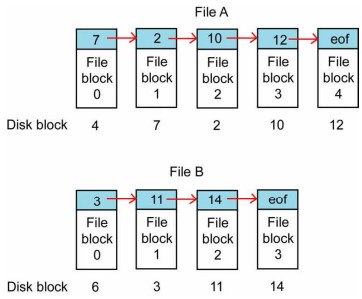


Souvislá alokace dat. bloků (2)

- **Výhody**
  - **Jednodušší implementace**.
  - Pro přístup k obsahu souboru musíme znát pouze:
    - diskovou adresu prvního dat. bloku,
    - počet dat. bloků alokovaných pro soubor.
  - **Výborný výkon čtení/zápisu**.
- **Nevýhody**
  - **Složitější alokace dat. bloků** (obvykle naznáme maximální velikost souboru při jeho vytváření).
  - **Fragmentace** disku.

Alokace dat. bloků pomocí zřetězeného seznamu

- Každý dat. blok obsahuje data a ukazatel na následující dat. blok.

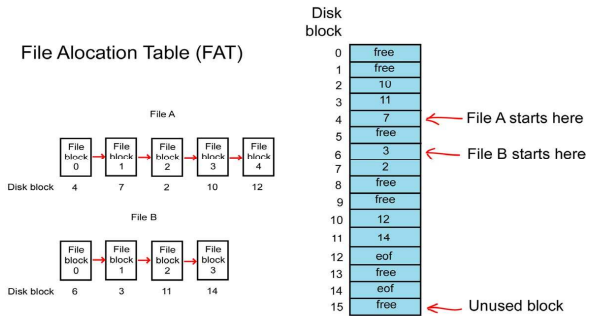


Alokace dat. bloků pomocí zřetězeného seznamu (2)

- **Výhody**
  - Pro přístup k obsahu souboru musíme znát pouze diskovou adresu prvního dat. bloku.
  - **Žádná fragmentace**.
- **Nevýhody**
  - **Pomalý náhodný přístup**.
  - Množství **dat v dat. bloku není přesně mocninou dvou.**

Alokace dat. bloků pomocí tabulky

- Alokace dat. bloků je založená na **zřetězeném seznamu**, ale **ukazatelé** na následující dat. blok **jsou uloženy v tabulce FAT**.
- Při přístupu k FS je FAT nebo její část nahrána do hlavní paměti.

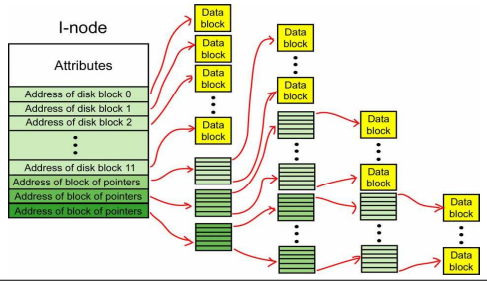


Alokace dat. bloků pomocí tabulky (2)

- **Výhody**
  - Celý dat. blok je využit pro uložení dat.
  - Náhodný přístup je rychlý (pokud je FAT v hlavní paměti).
  - Pro přístup k souboru stačí znát adresu prvního dat. bloku.
  - Informace o volných dat. blocích je obsažená ve FAT.
- **Nevýhody**
  - Velikost FAT (např. pro 20GB FS s velikostí dat. bloku 1KB, má FAT 20 milionů položek) .
  - Celá/část FAT musí být v hlavní paměti během používání FS.

Index nodes (I-nodes)

- i-node je struktura, která obsahuje jak **atributy souboru**, tak **adresy datových bloků**, ve kterých je uložen obsah souboru.
- Jsou tam **přímé adresy** a **nepřímé adresy** (první, druhé a třetí úrovně).
- **Speciální dat. bloky** jsou určeny pro uložení adres datových bloků při nepřímé adresaci.

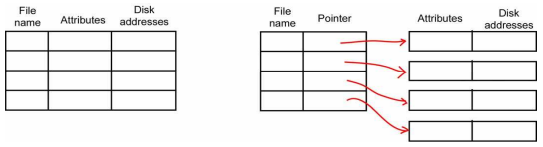


I-nodes (2)

- **Výhody**
  - V hlavní paměti jsou pouze i-nodes otevřených souborů.
- **Nevýhody**
  - Přístup k velkým souborům je pomalejší než k malým souborům.
  - Přibližně 10% datových bloků ve FS je použito na uložení adres dat. bloků (speciální dat. bloky).

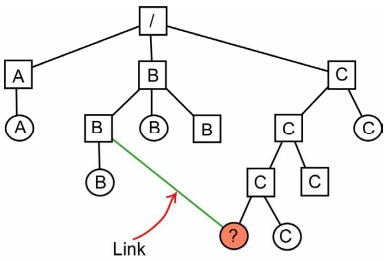
Implementace adresářů

- **Struktura položky adresáře**
  1. **jméno souboru, atributy souboru a adresy dat. bloků souboru**
     (FAT12, FAT16, FAT32).
  2. **jméno souboru a ukazatel na speciální strukturu**, která obsahuje atributy souboru a adresy dat. bloků souboru (např. UNIX).



Sdílené soubory

- Sdílené soubory mohou **být viděny současně** z různých adresářů pod různými jmény.
- **Změny obsahu souboru** v jednom adresáři by měli být **viděny ve všech adresářích**.
- **Link** = spojení mezi sdíleným souborem a adresáři.



Sdílené soubory (2)

- **Problém**
  - Nechť v adresářích jsou uloženy i adresy dat. bloků souboru.
  - Pokud jeden uživatel zvětší sdílený soubor o jeden dat. blok, pak ostatní uživatelé tuto změnu neuvidí.
- **Řešení 1** (hard link)
  - Každému souboru přiřadíme strukturu, která bude obsahovat adresy dat. bloků daného souboru (např. i-node).
  - V adresáři bude ukazatel na tuto strukturu..
- **Řešení 2** (soft link)
  - Adresáře ukazují na systémový soubor typu „LINK“, který obsahuje cestu ke sdílenému souboru.

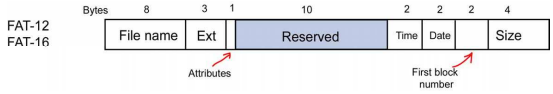
Příklad: MS-DOS

- Používá alokaci datových bloků **pomocí tabulky FAT (File Allocation Table)**
- **Jména souborů**
  - 8+3 velkých znaků: jméno+přípona (FAT-12, FAT-16),
  - 256 znaků (FAT-32).

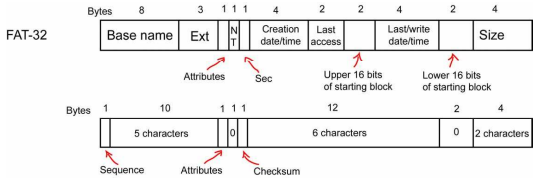
	Velikost diskové adresy [bit]	Velikost dat. bloku[KB]	Maximální velikost disk. oblasti
FAT-12	12	0.5 - 4	16 MB
FAT-16	16	2 - 32	2 GB
FAT-32	28	4 - 32	2 TB

Příklad: MS-DOS (2)

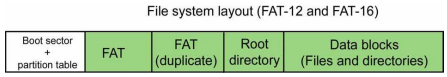
- **Položka adresáře ve FAT-12 a FAT-16** (32 Bytes)



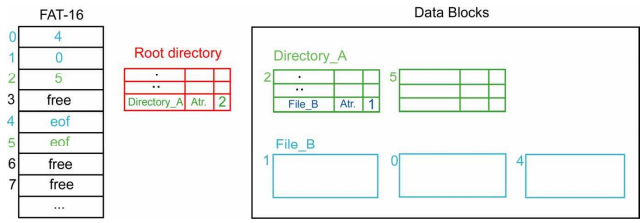
- **Položka adresáře ve FAT-32** (32 Bytes)
  - Každý soubor může mít dvě jména (jméno 8+3 a dlouhé jméno).
  - Dlouhé jméno je uloženo ve více položkách adresáře.



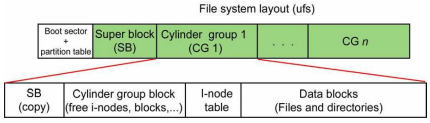
Příklad: MS-DOS (3)



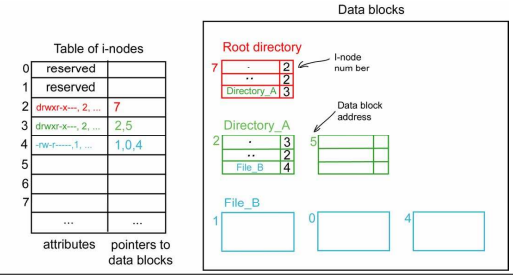
- **Přístup k souboru:** /Directory_A/File_B



Příklad: UNIX

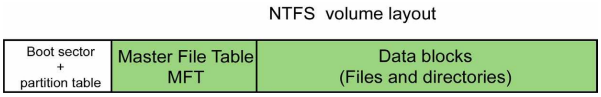


- **Přístup k souboru:** /Directory_A/File_B



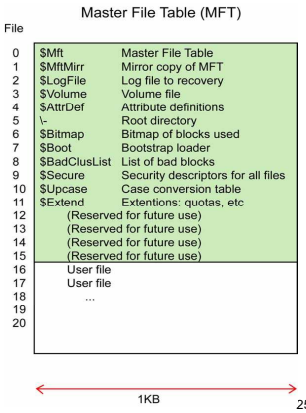
Příklad: NTFS

- **Jméno souboru**
  - cesta (32767 znaků) + jméno (255 znaků) v UNICODE.
- 64 bitové diskové adresy.
- Datový blok (Cluster): 512B,…,64KB.
- Podpora hard linků i symbolických linků.
- Umožňuje kompresi a kryptování.



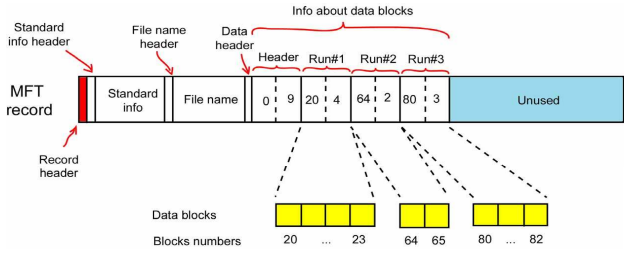
Příklad: NTFS (2)

- **Master File Table (MFT)**
  - Položka v MFT popisuje jeden soubor/adresář.



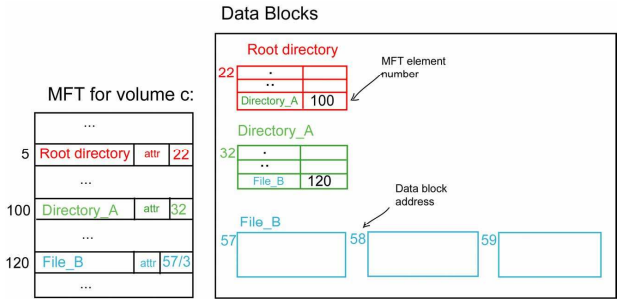
Příklad: NTFS (3)

- **Položka MFT**
  - se skládá z posloupnosti dvojic (attribute header a hodnota).
  - **rezidentní atribut** = attr. headrer i hodnota jsou v položce MFT.
  - **nerezidentní atributy** = attr. headrer je v položce MFT, ale hodnota je uložena v datových blocích.



Příklad: NTFS (4)

- **Přístup k souboru:** C:\Directory_A\File_B



Výkonnost FS

- Čtení dat. blok z disku je výrazně pomalejší než čtení slova z hlavní paměti.
- **Vyrovnávací paměť (block cache)**
  - Obsahuje některé informace načtené z FS na disku.
  - Část RAM.
  - Statická cache (fixní velikost, např. 10% RAM).
  - Dynamická cache (proměnná velikost, např. od 5 do50 % RAM).
- Pokud není místo ve vyrovnávací paměti, některé dat. bloky musí být odsunuty zpět na disk.
- Pro výběr dat. bloků, které odsuneme, používáme **modifikované algoritmy pro náhradu stránek** (FIFO, LRU, second chance).

Výkonnost FS (2)

- Strategie zápisu modifikovaných dat. bloků:
  - **Write through cache** (MS-DOS)
    - Všechny změny ve FS se zapisují okamžitě jak do vyrovnávací paměti tak do FS na disku (bezpečné, ale málo efektivní).
  - **Nonwrite through cache** (UNIX)
    - Čtení z vyrovnávací paměti je synchronní (data jsou současně načtena z FS na disku do vyrovnávací paměti i do procesu).
    - Zápis obsahu vyrovnávací paměti do FS na disku je asynchronní – prováděný periodicky pomocí systém. démona (např. **syncer** v UNIXu každých 30 sekund).

Žurnálované FS

- **Metadata**
  - důležité dat. struktury FS (super block, i-nodes, adresáře, ...).
- **Problém s tradičními FS**
  - Pokud při zápisu metadat na disk dojde k výpadku systému, FS může být v nekonzistentním stavu.
- **Řešení**
  - **Kontrolovat celý FS** po restartu systému (např. pomocí *scandisk* v MS Windows nebo *fsck* v UNIXu).
  - **Používat žurnálování u FS**.

Žurnálované FS (2)

- **Princip**
  - Ve FS je speciální soubor "**journal**", který obsahuje informace o transakcích na disku.
  - Při provádění disk. operace se nejdříve zapíše na disk informace o transakci, a potom se teprve operace začne provádět.
  - **Při výpadku FS**:
    - buď byla operace korektně dokončená a FS je konzistentní,
    - nebo se operace nedokončila, pak na základě „jurnalu" opravíme stav FS do konzistentního stavu (dokončíme nebo zrušíme operaci).
- **FS podporující jurnálování**
  - ntfs (MS Windows), ext3 (Linux), ufs (Solaris), jfs (AIX),...