

Operační systémy a sítě

Petr Štěpán, K13133

KN-E-229

stepan@labe.felk.cvut.cz

Téma 6. Správa zařízení a souborové systémy

Správa zařízení

- Non-volatile paměť (“nesmazatelná” paměť) paměť, která uchová informaci i bez napájení.
- Non-volatile paměť je určena pro dlouhodobé uchování informace.
- Příklady z historie:
 - Papírové děrné štítky.
 - ROM, paměť typu flash, feromagnetické paměti
 - Magnetické pásky a disky(e.g. Pevný disk, diskety, a magnetické pásky)
 - Optické disky (CD, DVD, BlueRay)

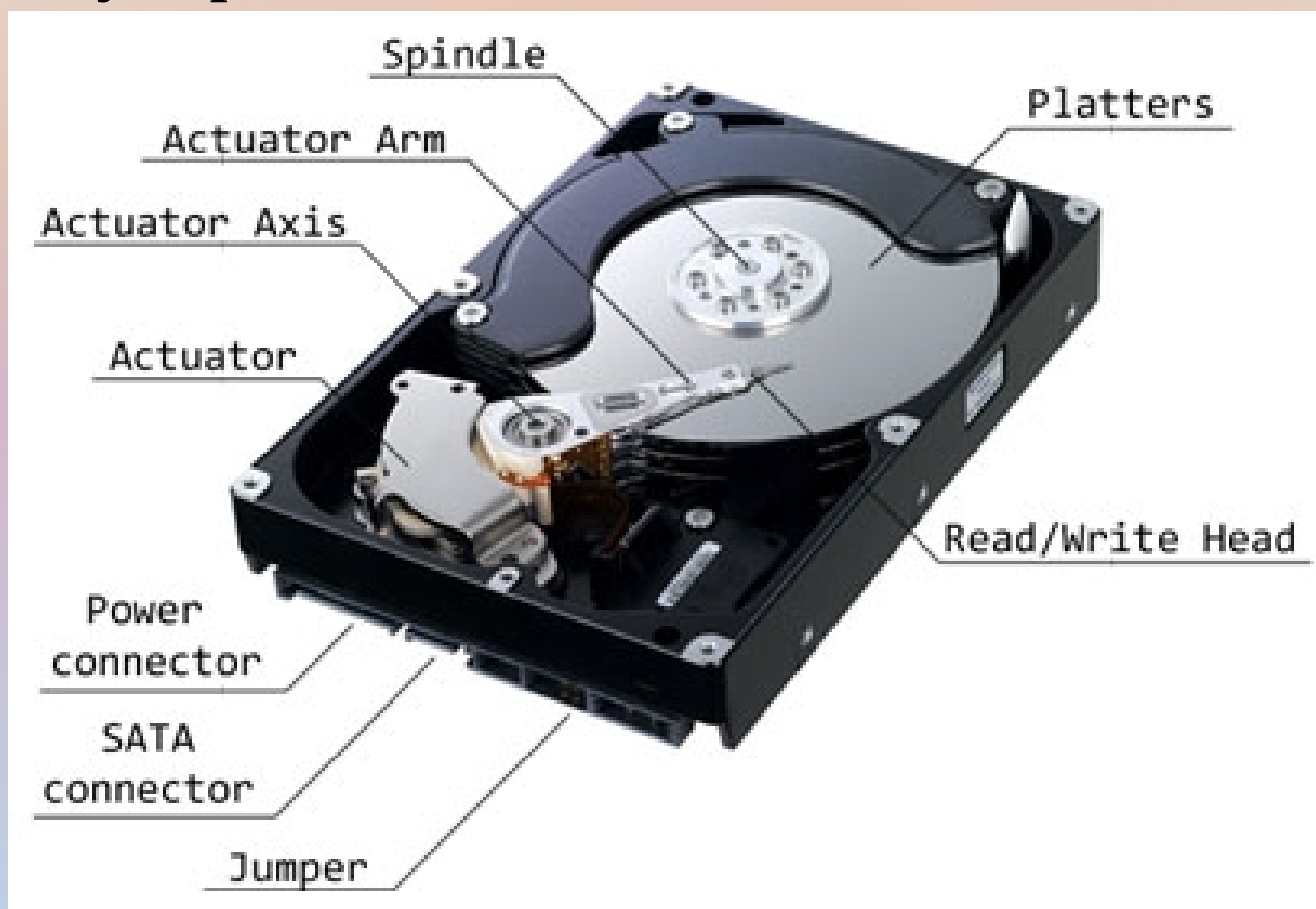
Magnetic tape data storage

- Poprvé použity 1951 pro uložení dat UNIVAC I
- Původně velikost kotoučů 10.5 palců, později zmenšená
- Přirozený sekvenční čtení a zápis
- Velmi vhodné pro zálohování dat i na delší čas
- Magnetická páska je výhodnější pro ukládání dat než optické disky
- Kapacita podobná HDD (5TiB v 2011)



Hard Disk Drive

- Vývoj od velikosti 8", přes 3.5" až 2.5", extrémě i 0.85"
Toshiba v roce 2004 4GB a 8GB verze
- Hlavička, případně hlavičky zapisují data na rotující disky s magnetickým povrchem



Hard Disk Drive

- Čtecí a zápisová doba je “stejná“
- Zpoždění čtení/zápisu závisí na:
 - Vyhledávací čas – posun hlavičky na správný cylindr (2-10 ms)
 - Rotační zpoždění – čekání na začátek sektoru, závisí na rychlosti otáčení (4.200 RPM – prům. 7.14ms, 7.200RPM – prům. 4.17ms, 15.000RPM – prům. 2ms)
 - Čtecí čas – čas na přečtení dat (0.2ms)
- Náhodné čtení 100KB/sec – kombinace všech tří čekání
- Náhodné čtení v jednom cylindru – 200KB/sec – pouze rotační a čtecí zpoždění
- Další sektor na stejném cylindru – 4MB/sec – nové disky i 140-400MB/sec

Solid-State Drive

- SSD nemá pohyblivé komponenty
- Odolný proti nárazům, tichý chod
- Většina SSD užívá NAND flash paměť, která udrží informaci i bez napájení
- Z konstrukčního hlediska je to RAM – random access memory – paměť s přímým přístupem
- Není rozdíl mezi sekvenčním a náhodným čtením
- Velký rozdíl mezi čtením a zápisem
- Čtení pod 100 μ sec
- Zápis je možný pouze do smazané paměti, mazání je zdlouhavé až 1.5 ms
- Pokud SSD má volné – smazané části, pak je zápis rychlý 200 μ sec. Jinak je nutné nejdříve paměť mazat 1.7ms

Solid-State Drive

- Mazání využívá vysoké napětí – omezená životnost buňek
- Paměťová buňka může být smazána 1k-100k krát, záleží na struktuře, SLC > MLC > TLC
- Firmware – organizační program od výrobce je zodpovědný za rovnoměrné využívání buňek
- Příkaz TRIM – OS může označit pro SSD části, které se nepoužívají
- Firmware je klíčovou částí SSD – porucha firmwaru (i HW) způsobí totální ztrátu dat
- Firmware zodpovídá
 - Mapování lineárního prostoru do SSD paměti
 - Rovnoměrné používání buněk
 - Udržuje smazané stránky pro rychlý zápis

Memristor

- Paměť blízké budoucnosti
- Paměť založená na nové součástce, která udrží informaci i bez napájení déle než flash paměti
- Rychlé čtení i zápis
- Dlouhý vývoj založený na nano-technologiích
 - 1971 – teoretická předpověď existence a praktický model
 - 2008 – první miniaturní realizace
 - 2013 – první memristorové součástky
- Seznam vědeckých článků na téma Memristor
<http://memlinks.eu/>

Souborové systémy

Co je to soubor?

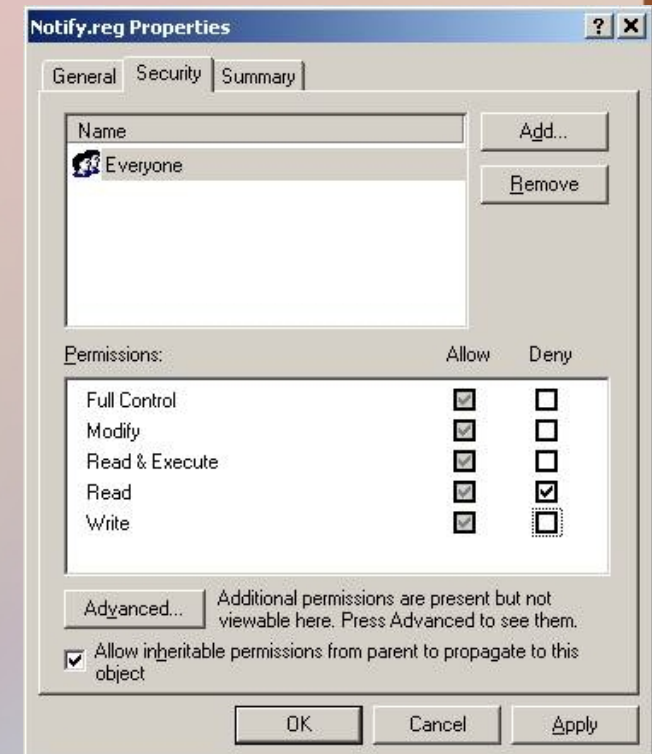
- Soubor je **pojmenovaná** množina dat
 - Původně jednorozměrná sekvence bajtů
 - První soubory byly ukládány na páskové paměti, které umožňovaly pouze jednorozměrný přístup
 - Soubor je dlouhodobější než program (persistentní struktura)
 - Data ze souboru jsou k dispozici i po ukončení programu
 - Struktura dat v souboru záleží na uživateli (programátorovi)
 - relativně volná – text, posloupnosti bajtů, ...
 - pevně formátovaná – přísně organizovaná data (záznam, blok, index, ...)
 - Soubory s jednoduchými záznamy
 - záznamy pevné či proměnné délky; řádky (proměnná délka s oddělovači)
 - Soubory s komplexní organizací
 - soubory s jednoduchými záznamy s vloženými řídicími strukturami
 - binární spustitelné soubory určené k zavedení a relokaci v paměti
 - soubory se záznamy uspořádanými do stromových struktur (indexy)
- Operační systém zajišťuje pro soubory
 - Vytvoření souboru s daným jménem a vlastnostmi
 - Otevření souboru, čtení a modifikaci otevřeného souboru
 - Uzavření souboru a uložení změn na paměťové médium
 - Správu adresářů

Příznaky souboru

- Vlastnosti (příznaky, atributy) souboru
 - **jméno**
 - jediná informace o souboru ukládaná v textové (uživatelé čitelné) podobě
 - **typ** souboru (též *manipulační typ*)
 - informace pro OS, jak s manipulovat s „daty“ obsaženými v souboru
 - adresář vidí OS jako množinu dat, ale pracuje s nimi spec. způsobem
 - **velikost**
 - okamžitá velikost souboru (zpravidla v bytech)
 - **umístění** (alokace)
 - souhrn informací o místech uložení obsahu souboru na sekundární paměti
 - **ochrany**
 - autorizační řídicí informace (kdo jak smí se souborem pracovat)
 - **vlastník**
 - identifikace vlastníka souboru (pro autorizaci)
 - **data a časy**
 - zpravidla čas **vytvoření**, poslední **modifikace** a posledního **přístupu** k souboru (pro správu a zálohování)

Ochrana souborů

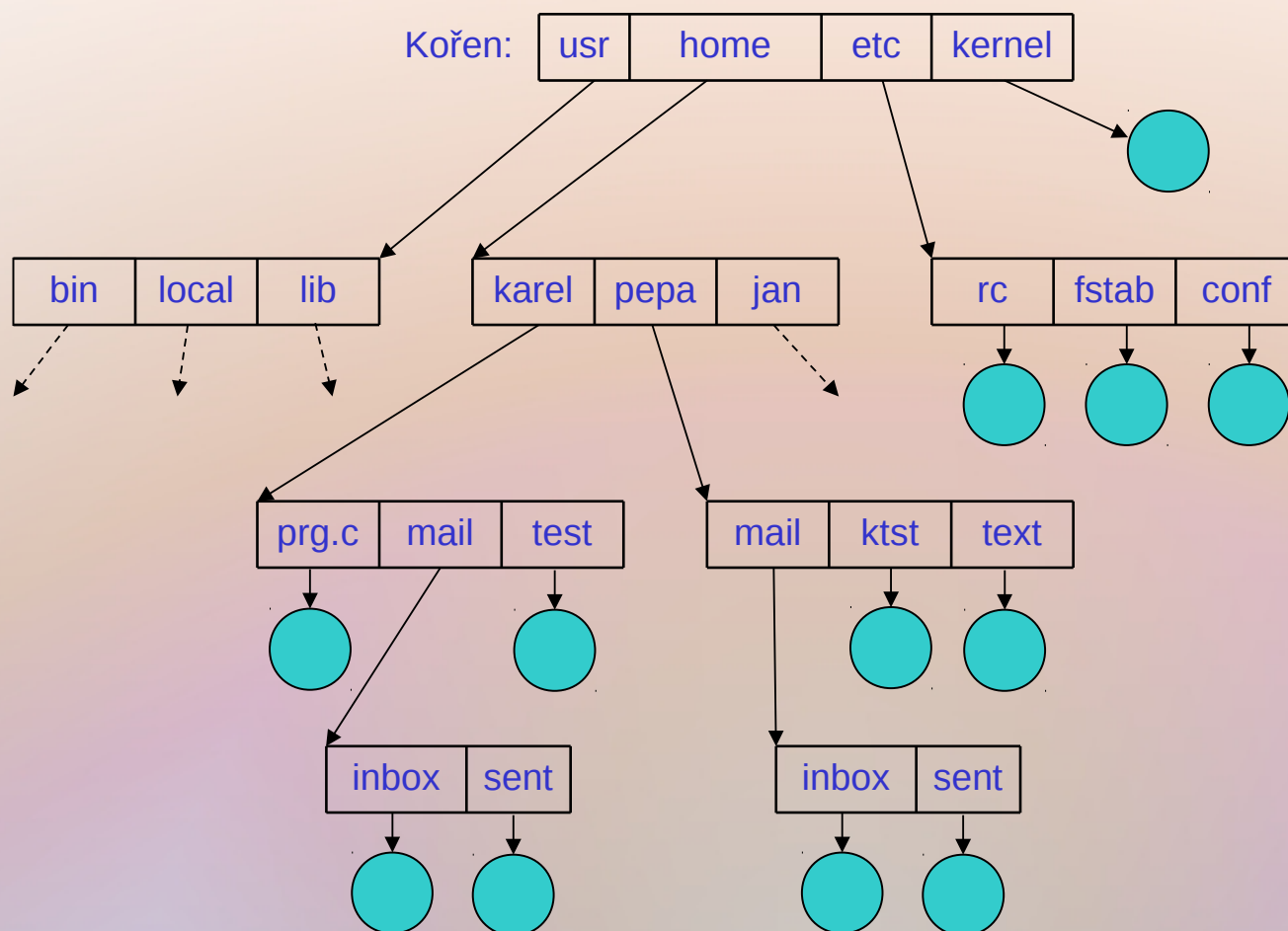
- Nutná (zejména) ve víceuživatelských systémech
- Volitelné řízení přístupu (*Discretionary Access Control*) – DAC
 - Vlastník souboru (obvykle ten, kdo ho vytvořil) má možnost určit, kdo smí se souborem co dělat
 - Typy přístupu **read**, **write**, **execute**, **append**, **delete**, ...
 - POSIX – bity **read**, **write**, **execute**; **user**, **group**, **other**
 - **rwX rwX rwX**
u g o
- Povinné řízení přístupu (*Mandatory Access Control*) – MAC
 - Možnosti práce se souborem určuje **systémová politika řízení přístupu** – pravidla (součást bezpečnostní politiky OS)
 - Běžní uživatelé systému nemají obvykle možnost pravidla měnit
 - může jen správce systému
 - např. Windows s NTFS



Adresáře

- Persistentní struktury k vyhledání pojmenovaných dat
 - Organizace souborů do adresářů
 - Adresář obsahuje „popisy“ souborů a případně i dalších adresářů
- Adresář
 - Množina datových položek uchovávajících informace o souborech uložených na sekundární paměti
 - obvykle na „diskovém oddílu“ →
 - Dvě pojetí pojmu „adresář“
 - 1) adresář souborového systému (nemusí obsahovat jména souborů)
 - 2) uživatelsky dostupná struktura se jmény souborů a odkazy do 1)
 - Položky adresářů obsahují atributy souborů
 - Operace s adresáři – potřebná **efektivita**
 - Vyhledání souboru, poskytnutí seznamu souborů
 - Vytvoření, zrušení či přejmenování souboru
 - Procházení souborovým systémem (změna pracovního adresáře)
 - Logická organizace adresářů
 - **seskupování** dle nějaké logické příbuznosti
 - **nezávislé pojmenovávání souborů**
 - 2 uživatelé mohou dát různým souborům totéž jméno
 - 2 uživatelé mohou pojmenovat týž (**sdílený**) soubor různými jmény
 - **struktury**: stromy, acyklické grafy, B-stromy

Adresáře se stromovou strukturou



- Položky v adresářích odkazují na jiné adresáře nebo na soubory (listy stromu)

Vlastnosti stromových adresářů

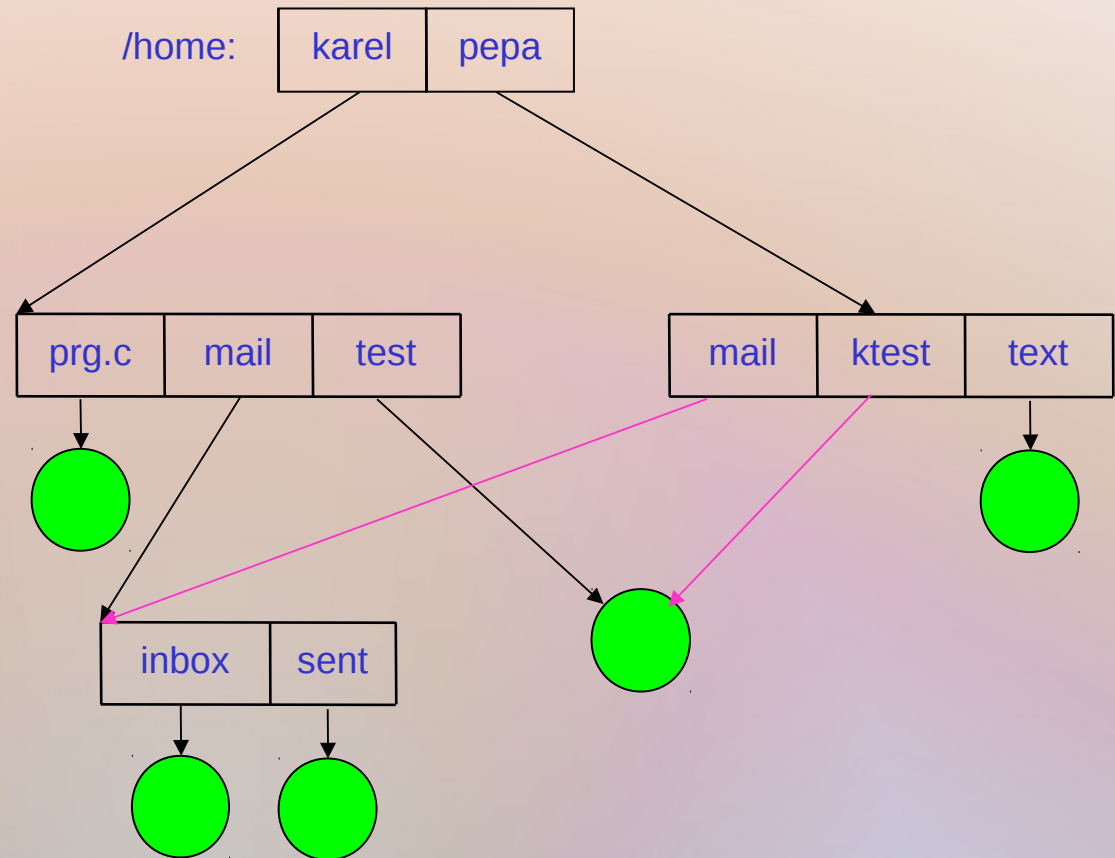
- Efektivní hledání
 - logaritmicky úměrné počtu souborů
 - aspoň průměrně, nebo při rovnoměrném zaplnění adresářů
- Nezávislé pojmenování
 - stejná jména pro různé entity
 - neumožňuje však různá jména sdílených souborů
 - vytváření a rušení souborů i adresářů všude, kde na to má uživatel právo
- Pracovní adresář a přístupová cesta k souboru
 - **Pracovní adresář**
 - dynamicky určovaný „výchozí bod“ v sadě adresářů
 - součást pracovního prostředí procesu
 - Úseky cesty – **oddělovač úseků**
 - POSIX / ; DOS, Windows \
 - Absolutní cesta – začíná v kořeni stromu
 - **/home/pepa/mail/inbox** – začíná oddělovačem úseků
 - Relativní přístupová cesta – vztažena k pracovnímu adresáři
 - Necht' **/home/pepa** je pracovní adresář, pak **mail/inbox** odkazuje totéž

Acyklické adresáře

- Umožňují sdílet soubory i adresáře
 - tzv. **aliasing** – jeden objekt má 2 či více různých jmen

- Problém:

- Zrušíme-li objekt `/home/karel/test`, bude nesmyslný odkaz `/home/pepa/ktest`



Dvě možná řešení

1. zpětné ukazatele

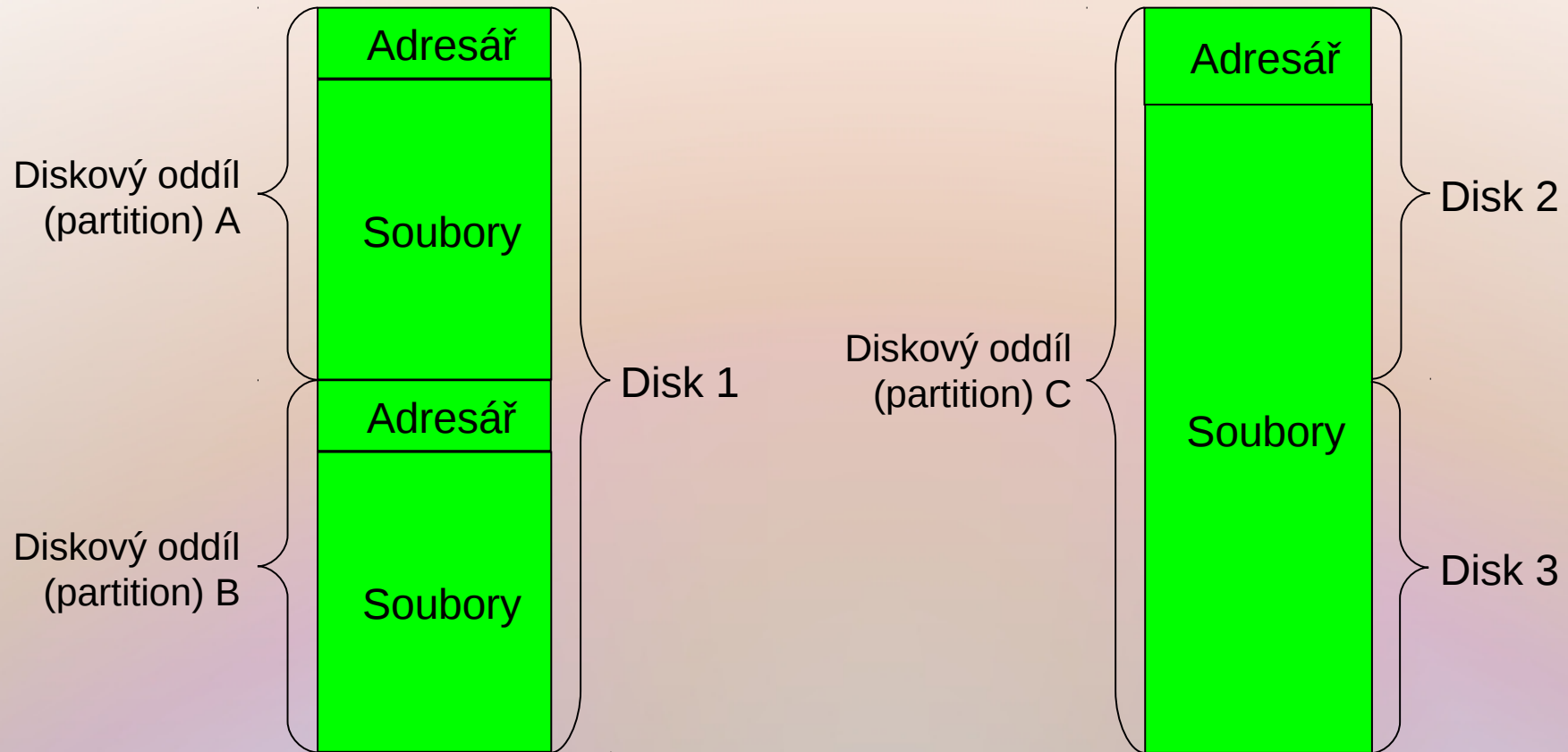
– objekt obsahuje údaj o místě, odkud naň vede odkaz

→ popisy objektů mají proměnnou délku

2. popisy objektů

obsahují čítače odkazů – objekt se fakticky zruší až když počet odkazů klesne na nulu (UNIX FS)

Organizace systému souborů

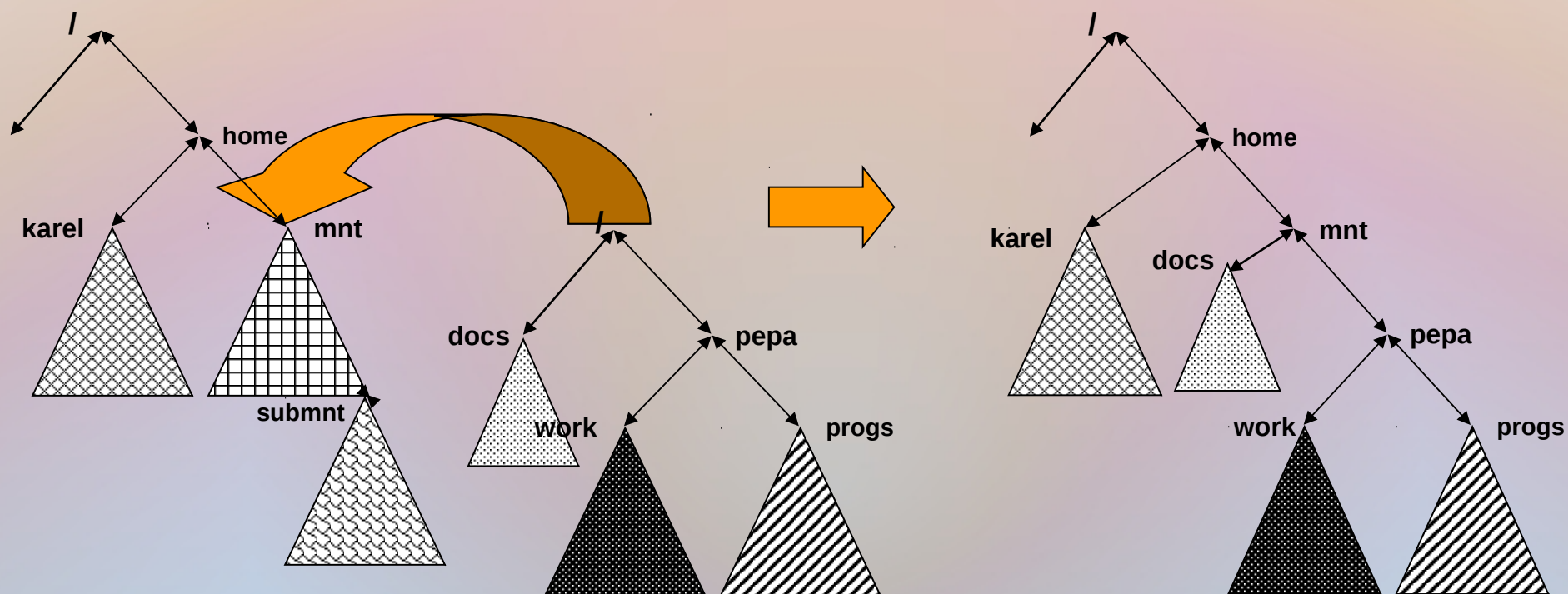


- Jeden disk je rozdělen na více **logických oddílů** a na každém z nich je samostatně organizovaný **systém souborů**

- Jeden **diskový oddíl** pokrývá více fyzických disků a **systém souborů** je vytvořen na tomto logickém oddílu

Připojování adresářových struktur

- Připojování souborových systémů (File system mounting)
 - Souborový systém na (výměnném, dosud nedostupném) mediu se musí zpřístupnit – připojit – **namontovat**
 - Připojuje se do udaného místa stávající adresářové struktury (mount point)
 - Dosavadní podstrom odkazovaný z místa, kam se montuje, přestane být dostupný



Základní operace se soubory

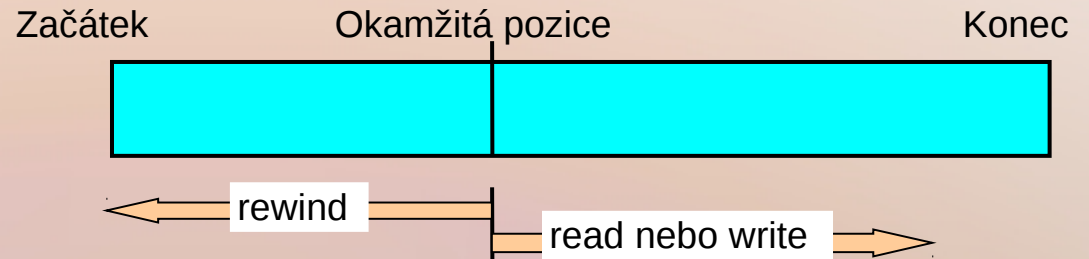
- POSIX
- Otevření souboru **fd = open(fn)**
 - vyhledání **záznamu o souboru** pojmenovaném **fn** v adresářových strukturách na sekundární paměti a přesunutí tohoto záznamu do hlavní paměti do **tabulky otevřených souborů**
- Uzavření souboru **close(fd)** - přesunutí **záznamu o souboru** z **tabulky otevřených souborů** na sekundární paměť
- Práce s obsahem souboru
 - **write, read** – tyto operace mění hodnotu ukazatele aktuální pozice v souboru, případně i obsah souboru
 - **seek** – změna pozice ukazatele v souboru
- Rušení souboru nebo jeho obsahu
 - **delete/remove** – zrušení souboru jako celku
 - v POSIX **unlink**
 - **truncate** – výmaz celého nebo části obsahu, zachová se existence souboru a jeho atributů

Přístupové metody

- Zpřístupňování záznamů v souboru
- Sekvenční přístup

– Standardní práce se souborem

- read
- write
- reset nebo rewind



- Přímý přístup

– **read n , write n**

- kde n je číslo záznamu
- OS zpravidla přímo nepodporuje

– **seek $d*n$, následované read nebo write**

– d je délka záznamu

– Určení záznamu jeho obsahem (klíčem)

| last name | logical record number |
|-----------|-----------------------|
| Adams | |
| Arthur | |
| Asher | |
| . | |
| . | |
| . | |
| Smith | 3 |

Index

| | | | |
|---|-------------|------------|-------|
| 1 | | | |
| 2 | | | |
| 3 | Smith, John | 7612071234 | 25000 |
| 4 | | | |
| . | | | |
| . | | | |
| . | | | |

"Datový" soubor

Tabulky otevřených souborů

- Tabulky otevřených souborů:
 - **tabulka procesu**
 - jedna pro každý proces, který soubor otevřel – co s otevřeným souborem proces dělá
 - **systémová tabulka** – co platí o souboru nezávisle na procesech
- Záznam o souboru v tabulce příslušné procesu
 - **Ukazatel** na právě **zpřístupňované místo** v souboru (*file pointer*)
 - **Přístupová práva** – podle způsobu otevření souboru
 - **Odkaz do systémové tabulky** otevřených souborů
- Záznam o souboru v systémové tabulce
 - **Čítač otevření** – kolikrát byl soubor otevřen (`open`), aniž byl zavřen (`close`) – záznam o souboru se odstraní z hlavní paměti, když čítač otevření klesne na 0
 - **Alokační informace** – umístění souboru na disku
 - **Velikost souboru** – zpravidla v bytech
 - **Časové údaje** – kdy byl soubor zpřístupněn, modifikován
 - **Zámky sdílení** – je-li soubor otevřen sdíleně

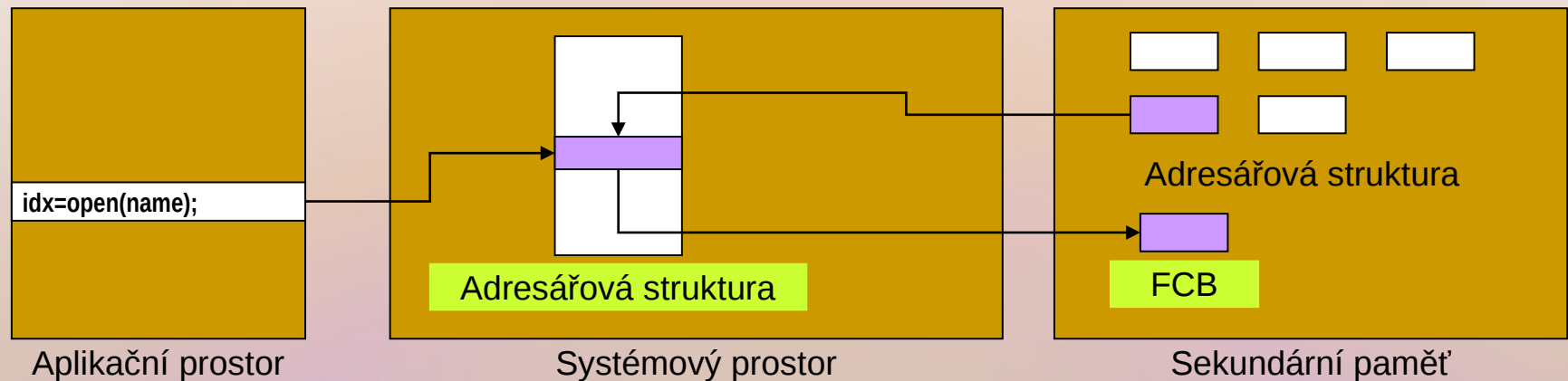
Implementace souborových systémů

- Systém souborů jako součást operačního systému bývá vrstven
 - **I/O Control**: drivery, správa přerušení
 - **Basic File System**: čtení/zápis fyzických bloků z/na disk
 - **File Organization Module**: správa (volné) paměti na disku
 - **Logical File System (LFS)**: správa metadat
 - organizace souboru, File Control Block – **FCB**, adresáře souborů, ochrany, bezpečnost
- **FCB – řídicí struktury pro práci se souborem**
 - Vytvoření souboru
 - Aplikace volá LFS, který vytvoří nový FCB, na disku opraví adresář a uloží nový FCB
 - Otevření souboru
 - LFS najde záznam o souboru na disku a jeho FCB zavede do paměti
 - LFS udržuje FCB otevřeného souboru v paměti
 - v systémové tabulce otevřených souborů

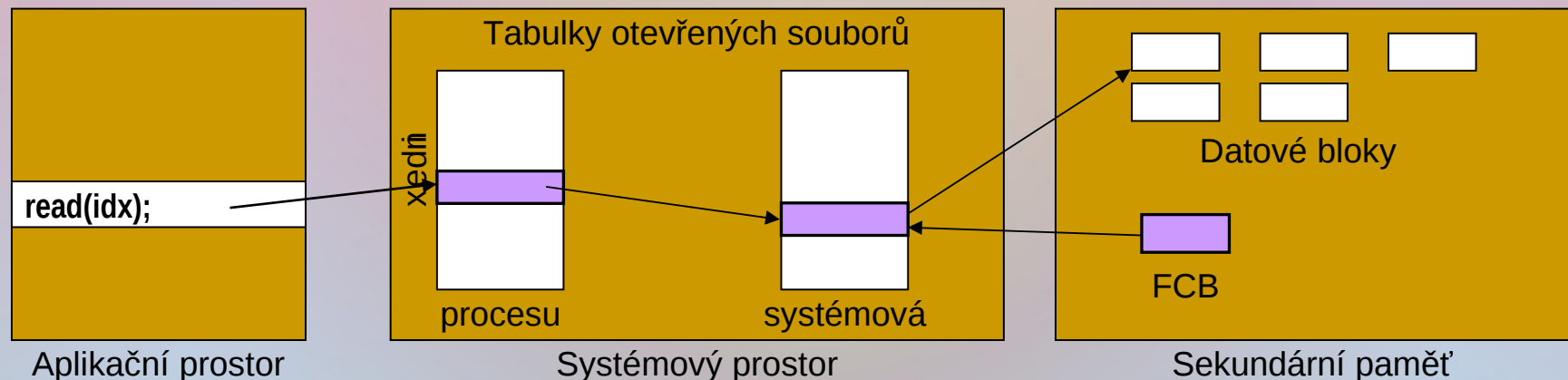
Datové struktury implementace FS

- Otevření souboru – jméno souboru se namapuje na tzv. index souboru

(manipulační údaj = **file-descriptor** – POSIX, **file-handle** – Windows)



- Čtení souboru

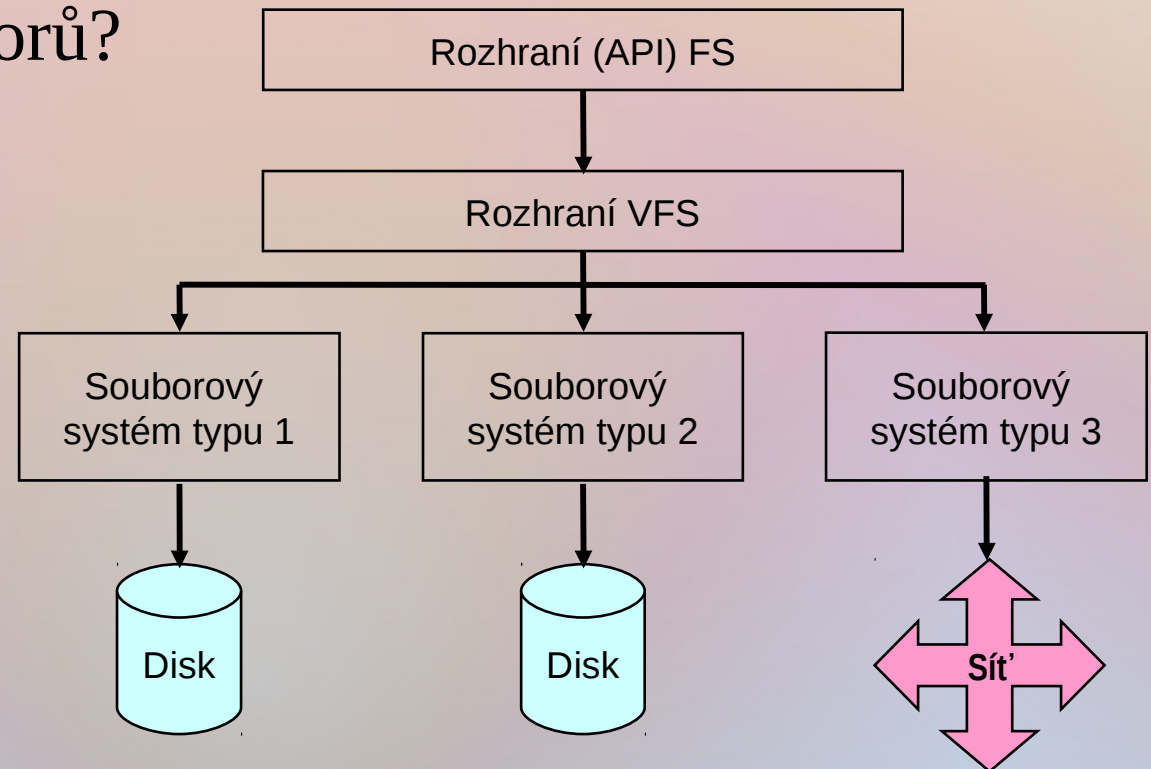


Čistý disk a disk spravovaný FS

- Raw disk – aplikace obhospodařuje prostor na disku
 - některé databázové systémy
- Disk spravovaný operačním systémem,
 - disk obsahující souborový systém
- Root (Boot) partition
 - obsahuje zaváděcí kopii OS
 - Boot Control Block
 - specifikace *root partition*
 - Unix: *boot block*
 - Windows: *partition boot sector*
 - Partition Control Block
 - specifikace datové oblasti (počet a rozměr bloků, odkaz na volnou paměť, odkaz na adresáře, ...)
 - Unix: *superblock*
 - Windows, NTFS: *Master File Table*

Virtualizace souborového systému

- Cíle virtuálního souborového systému (VFS)
 - možnost používat jednotné rozhraní systémových volání (API) i pro odlišné typy souborových systémů
 - API se vytváří spíše jako API k rozhraní VFS než jako rozhraní ke konkrétnímu systému souborů
- Proč více systémů souborů?
 - jiný pro pevné disky
 - jiný pro diskety
 - jiný pro CD, DVD, ...
 - interoperabilita různých OS



NFS, Novell, Win-Share=SMB 25

Implementace adresářů

- Musí se zajistit rychlé prohledávání
- Lineární seznam jmen souborů s ukazateli na bloky dat
 - jednoduše programovatelné, avšak vyhledávání souborů dle jmen je časově náročné
- Hašovaná tabulka – seznam s hašující strukturou
 - mohou se vyskytovat **kolize**, když různá jména generují tutéž adresu
 - vyžaduje se obvykle pevná velikost adresáře
- Komplexní datová struktura –
 - B+ strom NTFS v MS Windows
 - H-stromy – ext3/4 linux

Souvislé soubory

- Přidělování alokačních bloků souborům
- Přidělování souvislých diskových prostorů
 - Každý soubor zabírá množinu sousedních bloků disku
 - Obrovská externí fragmentace
 - Kolik bloků přidělit souboru, jehož velikost není předem známa?
 - Alternativa – Extent-Based File Systems
 - Souborům se přiděluje vždy několik souvislých úseků, tvořených několika diskovými bloky – *extents*
 - Soubor je tvořen jedním nebo více „extenty”
- Nesouvislé soubory
 - Vázané přidělování prostoru
 - Soubor je provázaným seznamem diskových bloků
 - Bloky jsou rozptýleny po disku libovolně
 - Pro každý soubor existuje uspořádaný seznam bloků, které soubor obsazuje
 - Indexované přidělování prostoru
 - Odkazy na bloky přidělené souboru jsou seskupeny v *indexovém bloku*, (též *tabulce indexů*)
 - Indexové bloky lze organizovat hierarchicky

Souvislé soubory

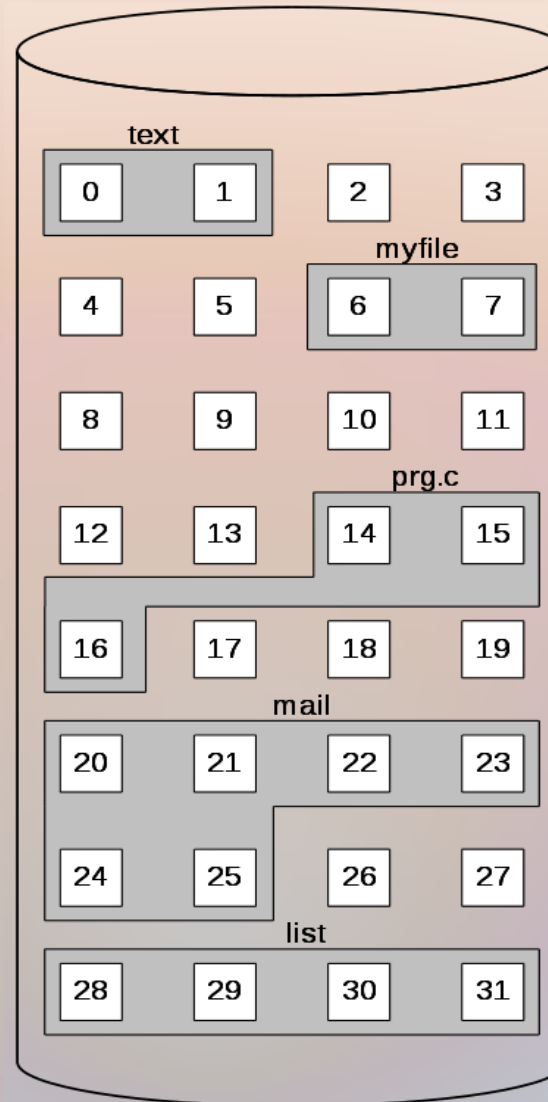
– Každý soubor zabírá
posloupnost sousedních
bloků

– Výhody

- Malé pohyby diskových hlav
– rychlé čtení
- Jednoduchá evidence – jen
začátek a počet bloků
- Sekvenční i přímý přístup

– Nevýhody

- Špatné využití diskového
prostoru
– hledání volného prostoru (BEST-
FIT, FIRST-FIT, ...)
- Soubory nemohou růst
(obtížné přepisování)
- Nutnost „setřásání“



Adresář

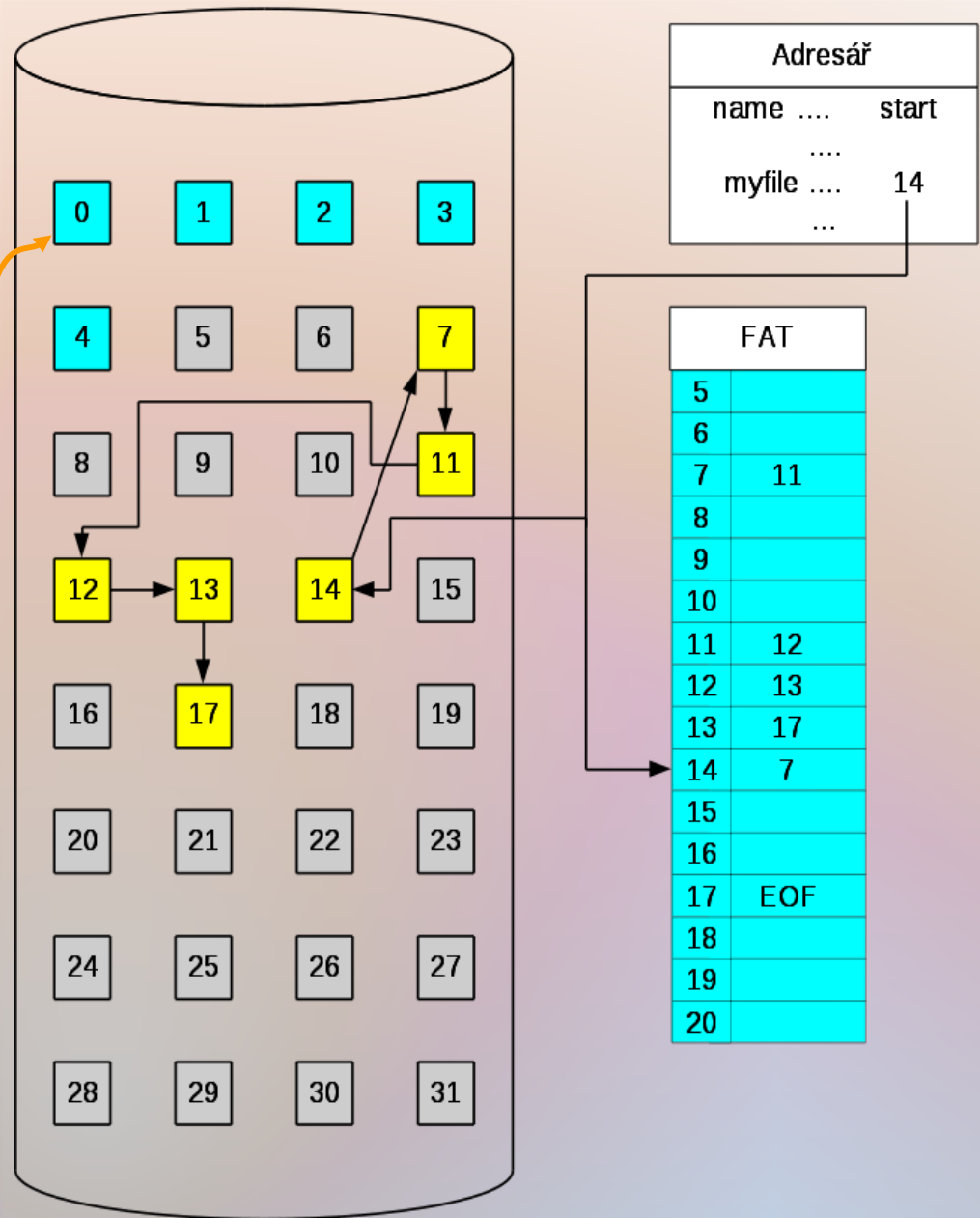
| <u>file</u> | <u>start</u> | <u>count</u> |
|-------------|--------------|--------------|
| text | 0 | 2 |
| myfile | 6 | 2 |
| prg.c | 14 | 3 |
| mail | 20 | 6 |
| list | 28 | 4 |

Vázané přidělování prostoru

- Alternativní názvy: **mapa disku**, *File Allocation Table (FAT)*
 - používáno v MS-DOS, OS/2, Windows 95/98/ME, ...
- Jednoduché
 - stačí znát jen adresu 1. bloku souboru
 - řetězený seznam bloků souboru
- Není nutno udávat velikost souboru při jeho vytváření
- Vhodné zejména pro sekvenční přístup
 - snadné přepisování
- Nevzniká externí fragmentace
 - netřeba setřásat
 - Přesto se to občas dělá kvůli přístupové rychlosti
- Problém s velikostí tabulky
 - velký disk
 - mnoho malých bloků → obrovská tabulka
 - méně velkých bloků → malé využití vlivem vnitřní fragmentace
 - nutný kompromis

Mapa disku a FAT

- **Mapa disku** – tabulka FAT je umístěna mimo vlastní oblast souborů na disku
- První blok souboru je odkazován z adresáře
- Další bloky jsou pak ve formě „rozptýlené tabulky“ uvedeny ve FAT
- Rezervované hodnoty ve FAT určují
 - konec řetězce bloků
 - vadné bloky
- Umístění FAT:
 - Konvencí určené místo na disku odkazované z „*Partition Control Block*“



Problém velikosti alokačního bloku FAT

- Alokační blok, **cluster**
 - posloupnost sousedních sektorů
- Fixní velikost FAT na disku
- Různé typy FAT
 - Položka ve FAT má velikost 12, 16 nebo 32 bitů
 - Tvar adresářové položky (MSDOS):

| | | | | | | | |
|--------|--------|---------|----------|-------------|-------------|---------|-------|
| FAT-16 | 8 bytů | 3 | 1 | 10 | 4 | 2 | 4 |
| | Jméno | Přípona | Atributy | Rezervováno | Datum a čas | 1. blok | Délka |

Adresační schopnost různých typů FAT

| Velikost sektoru | FAT12 | FAT16 | FAT32 |
|------------------|-------|--------|-------|
| 512B | 2MiB | 32MiB | 2TiB |
| 1KiB | 4MiB | 64MiB | 4TiB |
| 4KiB | 16MiB | 256MiB | 16TiB |

POZOR na velikost FAT tabulky, pro malé sektory a velké disky bude neúměrně velká

Windows - FAT-32

- Velké disky, dlouhá jména (UNICODE), zpětná kompatibilita
Základní adresářová položka

| | | | | | | | | | | |
|--------|---------|---|--------|---------|-----------------------|--------------------------|-------------------------|---------------------------------|-------------------------|------------------------|
| 8 bytů | 3 | 1 | 1 | 1 | 4 | 2 | 2 | 4 | 2 | 4 |
| Jméno | Přípona | A | N T | FC T | Datum a čas vytvoření | Poslední přístup (datum) | 1. blok horních 16 bitů | Datum a čas poslední modifikace | 1. blok dolních 16 bitů | Délka souboru v bytech |

(Fine Creation Time) – 1 byte s hodnotou 0 – 199 upřesňující čas vytvoření v 10 ms jednotkách.

Doplňková adresářová položka

| | | | | | | | |
|---------------|----|---|----|-----------------------|----|---------|---|
| 1 | 10 | 1 | 1 | 1 | 12 | 2 | 4 |
| 5 znaků jména | A | 0 | CS | 6 UNICODE znaků jména | 0 | 2 znaky | |

Sekvenční číslo

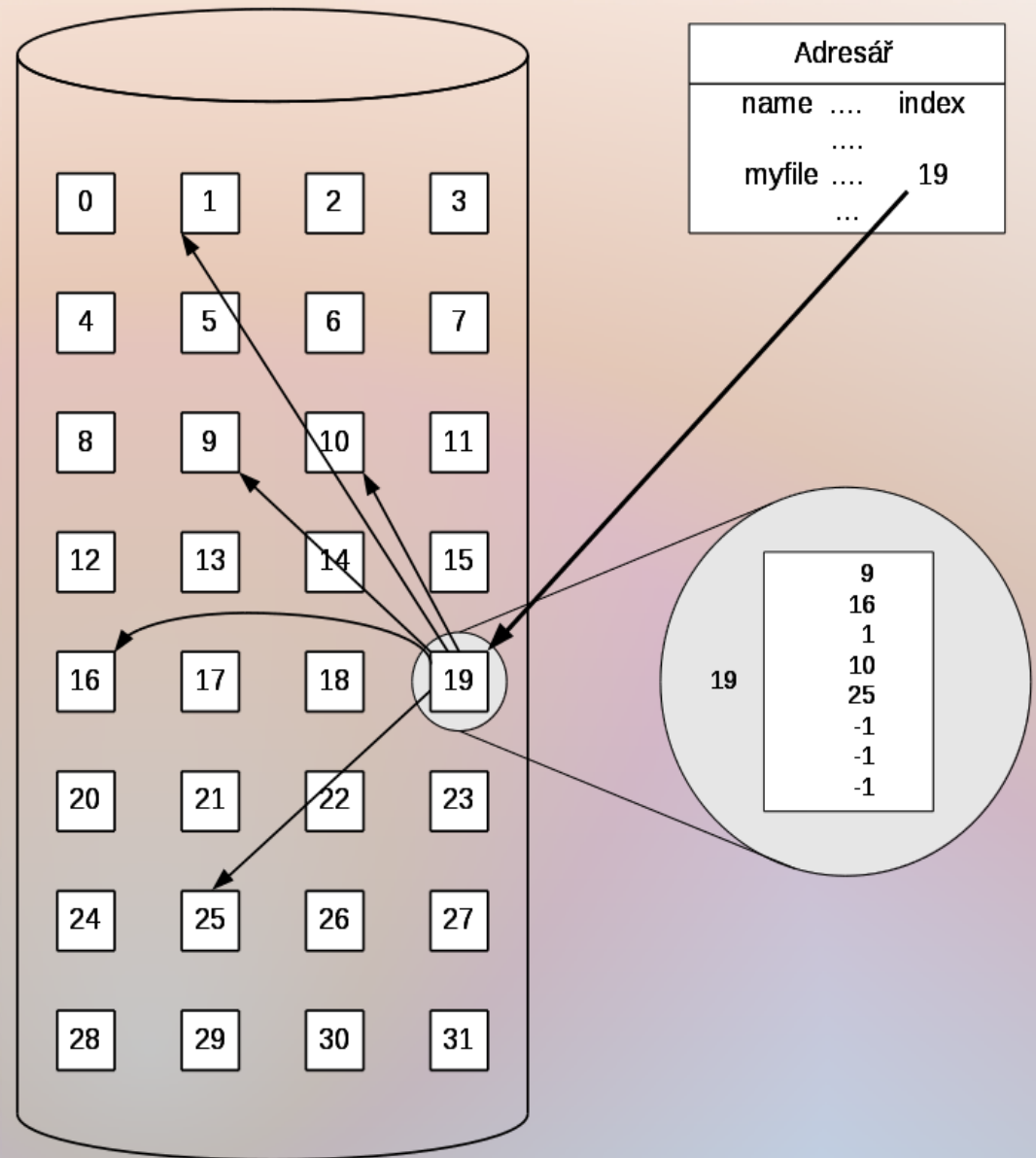
Kontrolní součet

Příklad: SouborUseUstrasneUmocUdlouhymUnazvemUcislo1.doc

| | | | | | | | | | | | | | | | | | | |
|----------|---|-----|---|---|---|---|-----------------------|----|------------------|---|----------------|---|------------------------|---|----------------|---|------------------------|--|
| 68 | s | l | o | 1 | . | A | 0 | CS | d | o | c | Ø | | | 0 | | | |
| 3 | h | y | m | | n | A | 0 | CS | a | z | v | e | m | | 0 | c | i | |
| 2 | a | s | n | e | | A | 0 | CS | m | o | c | | d | l | 0 | o | u | |
| 1 | S | o | u | b | o | A | 0 | CS | r | | s | e | | s | 0 | t | r | |
| SOUBOR~1 | | DOC | | A | 1 | 0 | Datum a čas vytvoření | | Poslední přístup | | 1. blok 16 MSB | | Datum a čas modifikace | | 1. blok 16 LSB | | Délka souboru v bytech | |

Indexové přidělování

- Položka adresáře odkazuje na blok obsahující **index** – seznam bloků
- Vhodné pro sekvenční i přímý přístup
- **Indexní blok** se při otevření souboru nahraje do operační paměti
- Indexy možno organizovat hierarchicky (Unix FS – UFS)

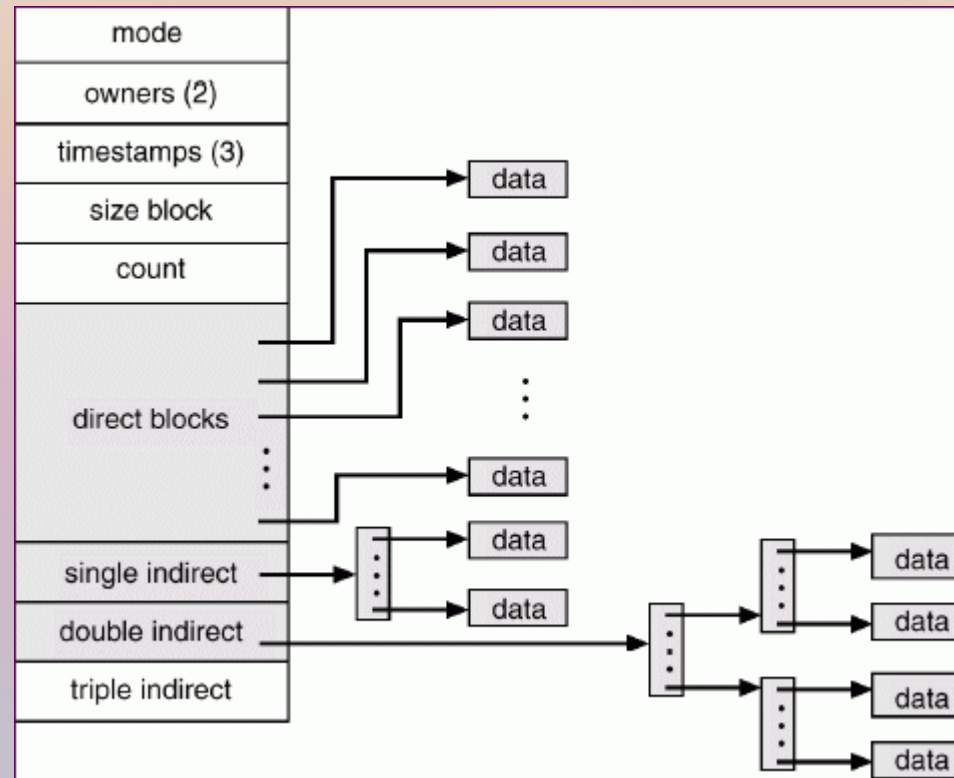


UNIX FS

Soubor je popsán tzv. **i-uzlem**

- v i-uzlu není jméno souboru
- i-uzly jsou odkazovány z adresářů

- i-uzel čítá odkazy vedené z adresářů a ruší se, když čítač klesne na 0
- i-uzel obsahuje informace o ochraně souboru



Správa volného prostoru

- Volná paměť jako řetěz volných bloků
 - analogie s tabulkou FAT
 - prostorově náročné
 - obtížné hledání souvislých bloků
- Bitová mapa (nejčastější)
 - co bit to diskový blok
 - bitová mapa umístěná na disku je úsporná
 - blok 4 KB = 2^{12} bitů, disk 64 GB = 2^{36} bytů = 2^{24} bloků,
 2^{24} bitů = 2^{16} bytů = 16 MB diskového prostoru – nepatrné %
 - bitová mapa se upravuje v paměti a nelze připustit, aby na disku se blok jevil jako volný, zatímco v paměti byl obsazen
 - okamžité propisování na disk
 - Řešení: nastav bit na disku, pak přiděl blok a pak teprve nastav bit v paměti

System souborů Windows NTFS

- Základní strukturou je svazek (*volume*)
 - Analogie *partition*
 - Na discích jsou svazky formátovány pomocí *disk administrator utility*
 - Svazek může být vytvořen na části disku, na celém disku nebo se může prostírat přes více disků
- Vše je popsáno jako tzv. Metadata – podobnost s databází
 - všechna metadata, vč. např. informace o svazku, jsou ukládána na disku jako soubory
- Struktura disku
 - ID sektor – Boot sektor
 - tabulka MFT
 - ostatní systémové soubory
 - oblast uživatelských adresářů a dat

System souborů Windows NTFS (2)

- Alokační blok (cluster)
 - navrženo i pro obrovské disky
 - Pro standardní 4KiB blok je max velikost disku 16TiB
- Vnitřní organizace NTFS
 - diskové adresy: pořadová čísla – *logical cluster numbers* (LCN)
 - soubor v NTFS
 - Není prostým proudem bytů jako v MS-DOS nebo v UNIXu
 - Jde spíše o strukturovaný objekt tvořeným atributy (**pojmenované atributy** – jméno, přístupová práva, doba vytvoření, ... + **bezejmenné atributy** – data)
 - je popsán jedním nebo několika záznamy v poli („řádku“) uchovávaném ve speciálním souboru („tabulce“) – **Master File Table (MFT)**

System souborů Windows NTFS (3)

- Zobrazení souboru
 - **rezidentní atributy** (definice a méně rozsáhlá data) uloženy přímo v **záznamech MFT**
 - **nerezidentní atributy** (nerezidentní vůči MFT) – rozsáhlé datové atributy v **externích alokačních blocích** referencovaných z rezidentních atributů
- Vlastnosti souborů
 - **vnější jméno** (až 255 UNICODE znaků)
 - jedinečné **vnitřní jméno, ID, file reference**
 - 64-bitový údaj tvořený dvojicí
 - 48-bitové **číslo souboru** (pořadové číslo definičního záznamu v MFT)
 - 16-bitové pořadové číslo inkrementované s každým použitím MFT záznamu (používá se pro vnitřní kontroly konzistence obsahu disku)
 - Prostor jmen NTFS je organizován do hierarchie adresářů
 - index jmen v každém adresáři má strukturu *B+*-stromu
 - v listech *B+*-stromu jsou vedle ukazatelů na data zopakovány atributy typu *jméno, velikost, doba vytvoření* (pro rychlé výpisy)
 - rychlé prohlížení – jména souborů jsou setříděná, doba prohledávání roste méně než lineárně s počtem souborů

NTFS: MFT a systémové soubory

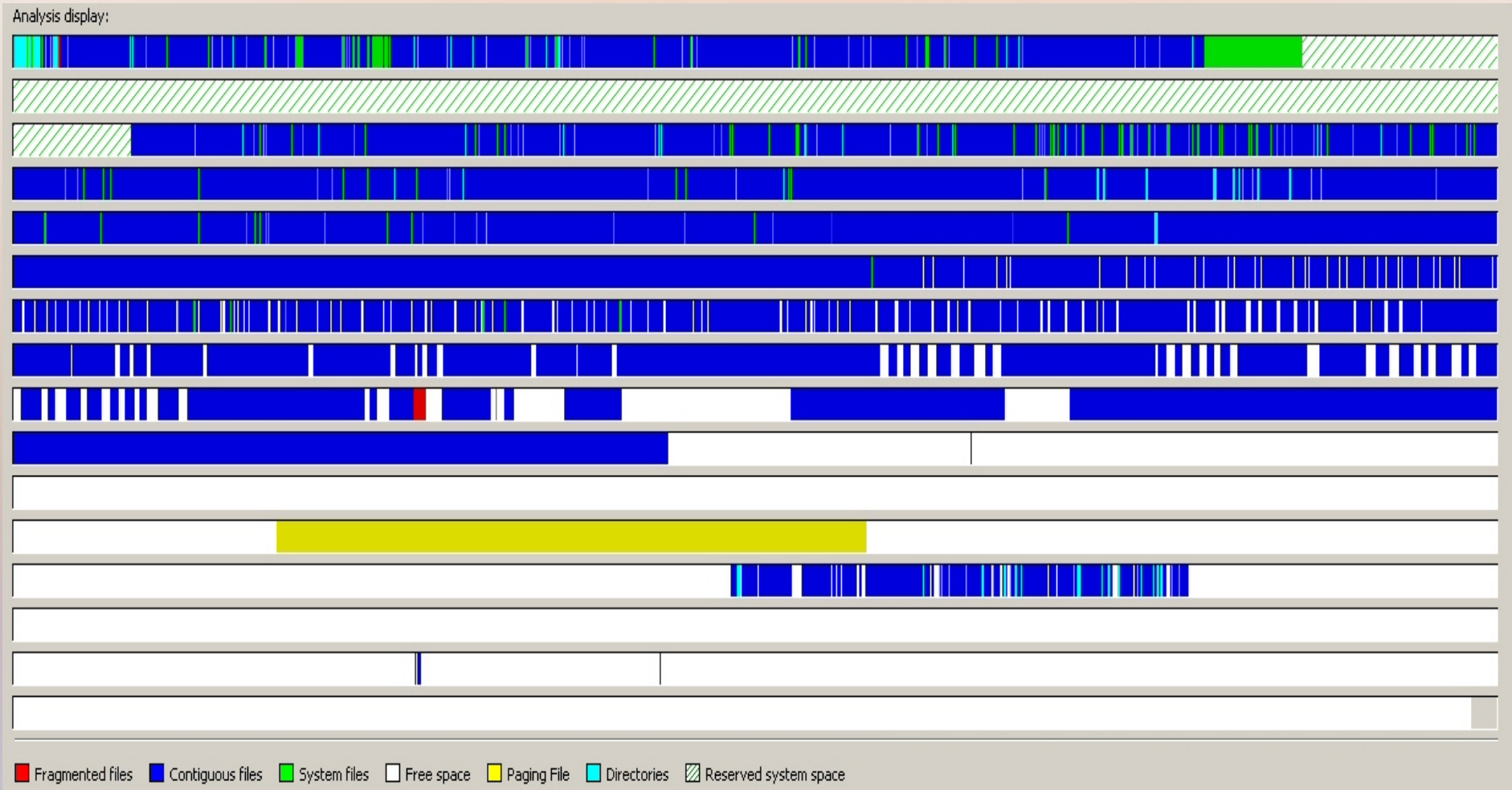
- Hlavní tabulka souborů, definice obsahu svazku
 - Relační databáze →
 - řádky (záznamy) – soubory, sloupce – atributy
 - záznamy MFT – definice souborů na NTFS svazku
 - komponenty záznamu MFT:
 - časová značka, čítač násobných vazeb, jméno souboru / adresáře, seznam externích alokačních bloků, bezpečnostní deskriptor (vlastník, kdo smí sdílet), data nebo index na data, bitová mapa použitých záznamů v MFT nebo v adresáři, ...
 - v MFT jsou záznamy s ukazateli na alokační bloky, které se nevešly do MFT struktury
- Systémové soubory
 - MFT a jeho záložní kopie
 - protokol: seznam akcí pro obnovu (*recovery*), změn adresářů, vytvoření souboru, ...
 - soubor se jménem svazku
 - soubor s definiční tabulkou atributů
 - soubor s indexem na kořenový adresář
 - soubor s bitovou mapou volných a přidělených alokačních bloků
 - soubor s definicí vadných sektorů

NTFS – zotavování z chyb

- Všechny korekce datových struktur systému souborů (**metadat**) jsou **transakce** → s protokolováním (*are logged*)
 - Dříve než se datová struktura změní, transakce zapíše záznam do protokolu, který obsahuje *redo* (jak operaci zopakovat) a *undo* informace (jak se vrátit do stavu před provedením operace)
 - Po změně datové struktury se do protokolu poznačí potvrzovací záznam (*commit record*) potvrzující úspěšné dokončení transakce
 - Když „systém spadne“, selže, lze datové struktury systému souborů pomocí záznamů v protokolu vrátit do konzistentního stavu z okamžiku před výpadkem systému
- Pomocí protokolovaných transakcí se řeší korekce systémových datových struktur, nikoliv korekce uživatelských souborů.
 - Není záruka obnovy všech uživatelských souborů po výpadku:
 - nesmí být porušeny soubory s metadaty
 - před výpadkem musí být systém v konzistentním stavu
- Protokol transakcí je uchováván jako metadatový soubor na počátku svazku
- Protokolování je v systémech Win 2000/XP realizováno službou *log file service*
 - tato „služba“ však není klasická *windows service* (není vidět jako proces)

NTFS - reálný snímek disku

- Snímek diskového oddílu o velikosti 60 GB



Porovnání koncepcí NTFS a FAT

- **NTFS** je určen pro disky s kapacitou větší než 500 MB
- **FAT** je pro stejný počet souborů méně paměťově náročný
- **FAT** má jednodušší strukturu, operace jsou efektivnější
- **NTFS** používá bezpečnostní deskriptor
 - individuální a skupinové řízení přístupu
 - ve FAT systému neexistuje
- Podpora **obnovy** je implementována jen v **NTFS**
 - seznam transakcí s daty
 - body regenerace (check-pointing) a automatická obnova konzistence
- **B+-stromová** struktura adresářů v **NTFS** – rychlejší přístup k souborům, minimalizace přístupů na disky, logaritmická složitost – průměrně $\log N$
- **FAT**: při hledání souborů vždy sekvenční průchod alokačními bloky adresářů – průměrně $N/2$
- Vytvoření souboru ve **FAT** systémech je **rychlejší**
- Otevření souboru ve **FAT**
 - je rychlé, je-li soubor na začátku adresáře
 - neexistuje-li soubor, je nutno prohledat celý adresář

Speciální soubory POSIX

- V POSIX systémech jsou všechna periferní zařízení považována za soubory
 - tzv. **speciální soubory**
 - i-uzel speciálního souboru je formálně shodný s i-uzlem diskového souboru
 - Místo alokačních informací jsou v i-uzlu dvě čísla
 - **major** – identifikuje ovladač ZVV, jehož prostřednictvím systém se ZVV komunikuje
 - **minor** – hodnota předávaná ovladači jako modifikátor jeho funkce. Obvykle udává, které z řady ZVV obhospodařovaných ovladačem i-uzel popisuje. Může obvykle svými jednotlivými bity zadávat ovladači doplňkové informace
 - Příklad:

| Jméno spec. souboru | Major | Minor | Význam |
|---------------------|-------|-------|---|
| /dev/ttyd0 | 20 | 0 | "Vstupní" modemová linka (call-in) |
| /dev/cua0 | 20 | 128 | Modem s telefonní volbou ven (call-out) |

Pseudosoubory POSIX

Vedle diskových a speciálních souborů považuje POSIX za soubory

- Symbolické spojky (*symbolic link, symlink*)
 - umožňují vést odkazy na soubory i přes jednotlivé diskové oddíly
 - cíl odkazu se nekontroluje
 - obdoba „zástupce“ (*shortcut*) ve Windows
 - služby OS umožňují použít symlink pro odkaz na soubor nebo symlink číst a měnit
- Roury (*fifos*) – pojmenované objekty pro lokální meziprocesní komunikaci
 - Z pohledu API se jako dvojice souborů chová i nepojmenovaný dočasný komunikační kanál zakládáný procesem za účelem komunikace jeho potomků. Pojmenovaná roura umožňuje, aby mohly komunikovat i procesy bez přímého společného rodiče (viz dále)
 - Spíše historická záležitost – nahrazeno sockety
- Sockety – pojmenované objekty pro komunikace po síti
 - Jeden proces socket otevře a „poslouchá na jeho výstupním konci“, jiné procesy mohou do „vstupního konce“ socketu posílat zprávy
 - Z pohledu API se jako dvojice souborů chová i nepojmenovaný dočasný socket – funkční rozšíření nepojmenované roury

Soubory v POSIX API

Každý nově spuštěný proces v POSIX-ovém systému zdědí od svého rodiče tři standardní soubory:

- **STDIN** – manipulační číslo 0 – soubor na němž se předpokládá základní vstup procesu – nejčastěji klávesnice spouštějícího terminálu
- **STDOUT** – manipulační číslo 1 – soubor na němž se předpokládá základní výstup procesu – nejčastěji obrazovka spouštějícího terminálu
- **STDERR** – manipulační číslo 2 – soubor, na němž se předpokládá chybový výstup procesu – nejčastěji obrazovka spouštějícího terminálu

Při zavírání souboru

- služba **close(fd)** způsobí, že manipulační číslo **fd** se uvolní pro další použití

Při otvírání souboru

- služba **fd = open(fname, ...)** použije nejmenší volné manipulační číslo **fd** uvolněné službou **close()**

Další důležité služby pro soubory POSIX

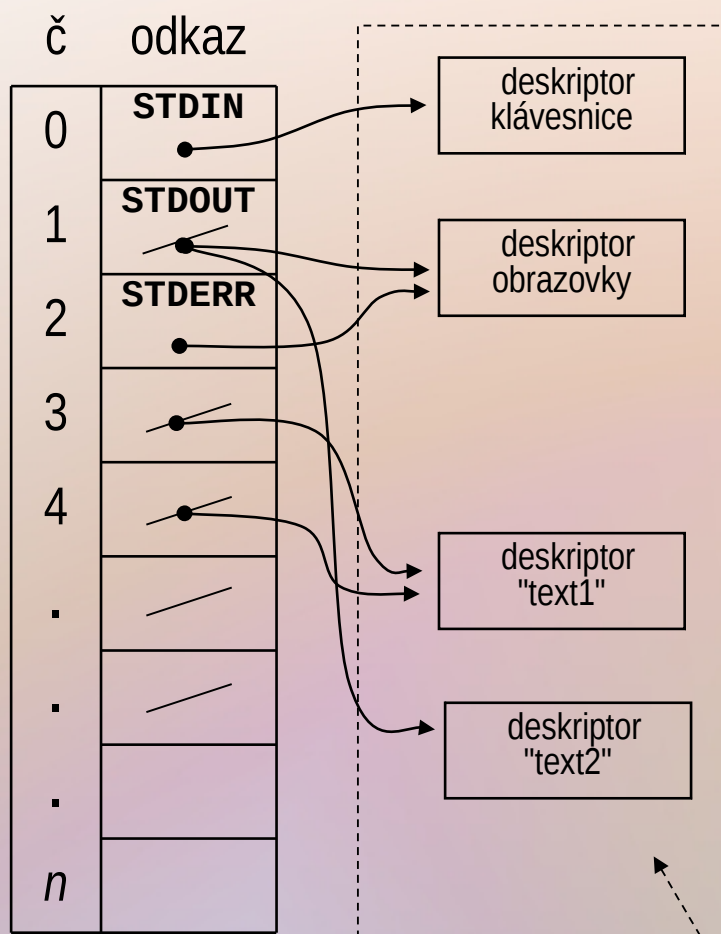
Vedle dříve vyjmenovaných POSIX služeb pro práci se soubory uveďme některé další:

- **fd = dup(fd0)** – duplikace manipulačního čísla souboru
 - Otevřený soubor přístupný přes manipulační číslo `fd0` je zpřístupněn i přes manipulační číslo `fd`, přičemž platí stejné pravidlo o přidělení tohoto čísla jako u operace `open()` (nejmenší volné)
- **int fd[2];**
s = pipe(fd) – založení komunikační **roury**
 - Vytvoří se komunikační kanál – **roura**. Služba vrátí do `fd[0]` manipulační číslo pseudosouboru, jehož prostřednictvím se zpřístupní „čtecí konec“ roury, a ve `fd[1]` je k dispozici manipulační číslo „zápisového konce“
 - Systém eviduje „počet otevření“, tj. počet odkazů na příslušný konec roury
 - Roura se automaticky zruší, jakmile čítače otevření na obou koncích klesnou na 0

Detaily uvedených služeb OS lze najít např. na

<http://www.freebsd.org/cgi/man.cgi?&sektion=2> a jejich použití na
<http://www.faqs.org/faqs/unix-faq/programmer/faq/>

Tabulka otevřených souborů z pohledu procesu



Velikost tabulky je systémově závislá a je součástí systémového popisu procesu

Výchozí stav, otevřeny pouze základní tři "soubory"

```
open("text1") → 3
```

```
close(STDOUT)
```

```
open("text2") → 1
```

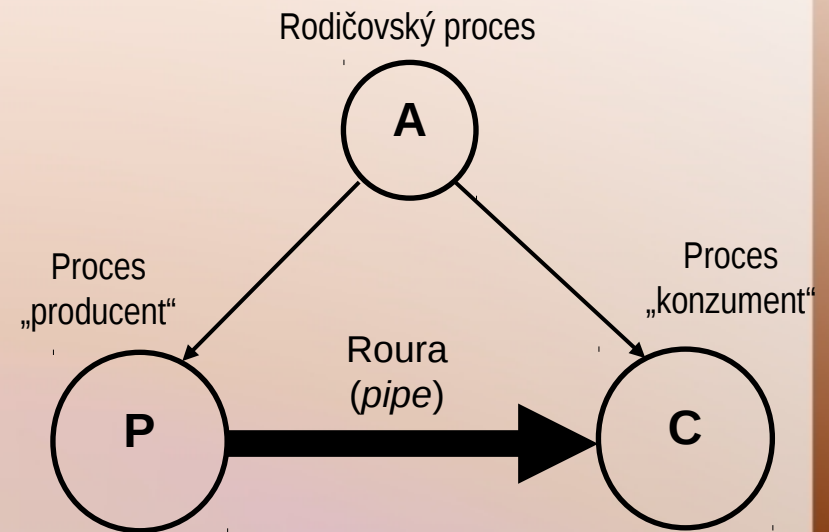
```
dup(3) → 4
```

Deskriptory jsou vytvářeny dynamicky v JOS podle potřeby

Implementace přesměrování

Uvedené služby pro práci se soubory umožňují implementaci přesměrování standardního vstupu či výstupu procesů

- Přesměrování obvykle zajišťuje rodičovský proces
- Rodič nejprve založí komunikační rouru službou `pipe()` a poté vytvoří své potomky službou `fork()`. Ti zdědí od svého rodiče všechny otevřené soubory včetně obou konců roury
 - Analogicky může rodič otevřít existující soubor pro vstup a/nebo vytvořit soubor výstupní
- V kódu potomků(a) provede příslušné manipulace s otevřenými soubory (zavírá a duplikuje manipulační čísla) a pak teprve volá službu `exec()`.



Implementace přesměrování – příklad

```
int pd[2];          /* pd[0] descr. čtení z roury    */
                   /* pd[1] descr. zápisu do roury   */
int child1, child2, wrval;
void main() {
    pipe(pd);       /* Vytvořit rouru    */
    child1 = fork();
    if(child1==0) { /* potomek 1 */
        close(1);  /* zavřít stdout    */
        dup(pd[1]); /* vstup do roury na stdout */
        close(pd[1]); /* zavřít nepoužité */
        close(pd[0]); /*      konce roury    */
        execl("./producent", "producent", 0);
    }
    /* původní rodič */
    child2 = fork();
    if(child2==0) { /* potomek 2 */
        close(0);  /* zavření stdin    */
        dup(pd[0]); /* výstup z roury na stdin */
        close(pd[1]); /* zavřít nepoužité */
        close(pd[0]); /*      konce roury    */
        execl("./konzument ", "konzument", 0);
    }
    /* původní rodič */
    close(pd[1]);  /* zavřít nepoužité */
    close(pd[0]);  /*      konce roury    */
    wrval = waitpid(-1); /* první potomek končí? */
    printf("Child%d finished\n", wrval==child1?1:2);
    wrval = waitpid(-1); /* druhý potomek končí? */
    printf("Child%d finished\n", wrval==child1?1:2);
    exit(0);
}
```

Roura se automaticky zruší, když na žádný z jejích konců nevede z žádného procesu odkaz

Tento kód neošetřuje chybové situace

To je dnes vše.

Otázky?