Operační systémy

Souborové systémy

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

I/O: zařízení

- zásadní složka Von Neumannova architektury
- různé pohledy na I/O zařízení: inženýrský (dráty, motory) vs. programátorský (rozhraní)
- různé rychlosti od 10 B/s (klávesnice) po 1 GB/s (PCI Express)
- různé druhy přístupu

Bloková zařízení

- data jsou přenášena v blocích stejné velikosti (typicky 512 B až 32 kB)
- možné nezávisle adresovat/zapisovat/číst data po jednotlivých blocích
- HDD, SSD, CD, DVD, páska?, ...

Znaková zařízení

- proud znaků/bytů (nelze se posouvat)
- klávesnice, myš, tiskárna, terminál

Ostatní

- nespadají ani do jedné z kategorií
- hodiny (přerušení), grafické rozhraní (mapovaná paměť)

Bloková zařízení: HDD

- disk – plotny, stopy ($\Longrightarrow$ cylindry), sektory (typicky 4 kB, dříve 512 B)
- původně se adresovaly sektory ve formě CHS (praktická omezení, mj. velikost disku)
- nahrazeno LBA (logical block addressing)
- low-level formát $\Longrightarrow$ hlavička + data + ECC
- připojené typicky přes (P)ATA, SATA
- rychlost přístupu ovlivňuje
  - nastavení hlavičky na příslušný cylindr (seek time; nejzásadnější)
  - rotace (nastavení sektoru) pod hlavičku
  - přenosová rychlost
- nezávislá cache (hromadí požadavky $\Longrightarrow$ eliminuje přesuny)

Odbočka: přístupové doby

L1 cache reference	0.5 ns	
Branch mispredict	5 ns	
L2 cache reference	7 ns	
Mutex lock/unlock	25 ns	
Main memory reference	100 ns	
Compress 1K bytes with Zippy	3,000 ns	
Send 1K bytes over 1 Gbps network	10,000 ns	(0.01 ms)
Read 4K randomly from SSD*	150,000 ns	(0.15 ms)
Read 1 MB sequentially from memory	250,000 ns	(0.25 ms)
Round trip within same datacenter	500,000 ns	(0.5 ms)
Read 1 MB sequentially from SSD*	1,000,000 ns	(1 ms)
Disk seek	10,000,000 ns	(10 ms)
Read 1 MB sequentially from disk	20,000,000 ns	(20 ms)
Send packet CA-Netherlands-CA	150,000,000 ns	(150 ms)

HDD: optimalizace

- víc požadavků se bude řešit najednou
- místo sektorů se pracuje s clustery sektorů (velikost podle velikosti disku)
- cache disku
- cache OS $\Longrightarrow$ společně s VM; cachuje se na úrovni FS
- odpovídající algoritmy – LRU, LFU, ..., jejich kombinace
- zjednodušení OS $\Longrightarrow$ otevření souboru namapování do cache; demand paging
- write-through cache: data se po zapsaní zapisují přímo na disk
- write-back cache: data se zapisují až po čase (možnost optimalizací zápisu)
- vynucení uložení cache (`flush`)
- sekvenční čtení
  - read-ahead – data se načítají dopředu
  - free-behind – proaktivně uvolňuje stránky, při načítání nových
- ,,spolupráce" – OS & HW (spoon-feeding); databáze

Bloková zařízení: RAID

- SLED: Single Large Expensive Disk
- RAID: Redundant Array of Inexpensive/Independent disks
- Mean Time to Failure: $MTTF_{pole} = \frac{MTTF_{disk}}{N}$
- hardware vs. software RAID
- RAID-0 (stripping): zvýšení propustnosti, problém selhání pořád existuje
- RAID-1 (mirroring): zvýšení propustnosti (kopie), řeší problém selhání
- RAID-2: dělí data na po bitech; Hammingův kód; disk pro paritu (napoužívá se)
- RAID-3: dělí data po bytech; XOR; disk pro paritu (zátěž); zvládne výpadek jednoho disku
- RAID-4: jako RAID-3 používá bloky (zátěž)
- RAID-5: jako RAID-4; paritní bloky jsou ale distribuovány
- RAID-6: jako RAID-5; Reed-Solomon kód; dva paritní bloky; výpadek až dvou disků
- kombinace: RAID-0+1, RAID-1+0

Bloková zařízení: SSD, CD, DVD

Solid-state Drives

- flash paměti; rozhraní jako HDD
- bez rotujících částí $\Longrightarrow$ rychlý přístup (výrazně víc IOPS)
- problematický zápis
  - omezení na počet přepsání jednoho místa
  - paměť musí být nejdřív vymazána
  - často lze zapisovat po stránkách, ale mazat je nutné po blocích $\Longrightarrow$ rychlější zápis než přepis
- wear levelling
  - *žádný* – data se přepisují na místě
  - *dynamický* – změněné bloky označeny jako neplatné a data zapsány jinam (USB)
  - *statický* – jako dynamický, ale přesouvá i nezměněné data (SSD)
  - hardwarová vs. softwarová implementace (JFFS2, LogFS)
- garbage collection + TRIM

Compact Disc (CD)

- data umístěna ve spirále $\Longrightarrow$ pomalé vyhledávání; rychlé sekvenční čtení
- vysoká redundance dat
- symbol - k zakodávání 8 b se používá 14 b
- 42 symbolů tvoří rámec o velikosti 588 b (192 b data, zbytek ECC)
- jeden sektor obsahující 2048 B dat je tvořen 98 rámci (zahrnuje 16 B hlavičku a 288 B pro ECC)
- efektivita 28%!

DVD, BR-D

- analogicky jako CD
- jiné vlnové délky laseru

Souborové systémy: Motivace

- potřeba uchovávat větší množství dat (primární paměť nemusí dostačovat)
- data musí být perzistentí (musí přežít ukončení procesu)
- k souborům musí být umožněn souběžný přístup
- $\Longrightarrow$ řešení v podobě ukládání dat na vnější paměť (např. disk)
- $\Longrightarrow$ data ukládána do souborů tvořící souborový systém (File System/FS)
- soubor jako proud bytů (doprovázen doplňujícími informacemi)
- souborový systém jako abstrakce (odstínění od implementačních detailů)
- $\Longrightarrow$ Unix

Operace se soubory (1/2)

- `create` – vytvoření souboru
- `write/append` – zápis do souboru (na konec, popř. přepis); souvislý bloky vs. postupný zápis
- `read` – čtení ze souboru (do přichystaného bufferu)
- `seek` – změna pozice
- `erase` – odstranění souboru (uvolnění místa); (`link` & `unlink`)
- `truncate` – zkrátí daný soubor na požadovanou velikost
- ne všechny souborové systémy (a zařízení podporují všechny operace); např. CD+ISO 9660
- operace dostatečně obecné $\Longrightarrow$ přístup k zařízením (disk, klávesnice, terminál); ovládání systému
- $\Longrightarrow$ lze používat existující nástroje pro práci se soubory
- např. využití při správě OS (`procfs`, `devfs`, `sysfs`)
- roury, FIFO

Operace se soubory (2/2)

- `open` – otevře soubor, aby s ním šlo manipulovat přes popisovač (file descriptor, file handle)
- ukazatel na strukturu v jádře
- přístup přes jméno neefektivní
- „soubory" nemusí mít jméno
- jeden soubor může být otevřen vícekrát (více ukazatelů na pozici v souboru)
- `close` – uzavře soubor
- `get/set attribute` – práce s attributy (metadaty)

Typy souborů

- běžné soubory, adresáře, (Unix: speciální soubory pro blokové a znakové zařízení)
- binární vs. ASCII soubory (ukončení řádků)

Organizace souborů (1/3)

- soubory jsou rozlišovány podle nazvů (často specifické pro daný OS nebo FS)
- rozlišování velkých a malých písmen (Unix vs. MS-DOS a Windows)
- MS-DOS: požadavek na jméno souboru ve tvaru 8+3 (jméno + přípona)
- rozlišení obsahu souboru
  - podle přípony (Windows–asociace s konkrétní aplikací)
  - magická čísla (Unix–podle úvodních bytů je možné identifikovat typ)
  - podle metadat (informace o souboru jsou uloženy vedle souboru, jako součást FS)
- typicky se soubory organizují do adresářů (složek)
- každý adresář může obsahovat běžné (popř. speciální) soubory i další adresáře $\Longrightarrow$ stromová struktura (používaly se i omezenější systémy – žádné, jedna, dvě úrovně)
- v zápisu cesty ve stromě se používá lomítek
  - Unix: `/usr/local/bin`
  - Windows: `\usr\local\bin`

Organizace souborů (2/3)

- k přístupu k souboru se používají
  - absolutní cesty `/foo/bar/baz.dat`
  - relativní cesty `foo/bar.dat` $\Longrightarrow$ každý proces má aktuální adresář
- operace s adresáři: Create, Delete, OpenDir, CloseDir, ReadDir, Rename
- speciální adresáře v každém adresáři „.“ a „..“ (aktuální adresář, nadřazený adresář) $\Longrightarrow$ nutné pro navigaci v hierarchii adresářů
- struktura nemusí být hierarchická $\Longrightarrow$ obecný graf (acyklický, cyklický)
  - **hardlink** – ukazatel na soubor (jeho tělo/obsah)
  - **symlink** – soubor je odkaz na jiný soubor (specifikovaný cestou)
- v Unixech běžné, ve Windows delší dobu (ale chybělo rozhraní)

Dělení disku

- každý fyzický disk se skládá z jedné nebo více logických částí (partition; oddíl); popsané pomocí partition table daného disku
- v každé partition může existovat souborový systém (označovaný jako svazek)
- v Unixech je každý svazek připojen (mounted) jako adresář (samostatný svazek pro /, `/home`, `/usr`)
- $\Longrightarrow$ Virtual File System (VFS)
  - využití abstrakce $\Longrightarrow$ umožňuje kombinovat různé FS do jednoho VFS
  - specializované FS pro správu systému (`procfs`, `sysfs`) $\Longrightarrow$ API OS
  - možnost připojit běžný soubor jako svazek (i svazek je soubor)
  - síťové disky (NFS, CIFS)
- ve Windows jednotlivé svazky označeny (a:, b:, c:, . . .); ale funguje i mountování (preferovaný jeden svazek pro vše)

Struktura souborů

- často je soubor chápán jako proud bytů
- sekvenční vs. náhodný přístup
- někdy může být výhodná jiná struktura
- rozdělení jednoho souboru na více proudů (např. spoustitelný soubor – kód + data)
- potřebná podpora ze strany FS i OS $\Longrightarrow$ streamy v NT
- společně s daty jsou k souboru připojena metadata (atributy)
  - vlastník souboru
  - přístupová práva
  - velikost souboru
  - příznaky (skrytý, archivace, spustitelný, systémový)
  - čas vytvoření, čas posledního přístupu

Přístup k souborům

Zamykání

- sdílený přístup
- omezení přístupu – současné čtení (zabránění zápisu)

Sémantika konzistence

- chování systému při současné přístupu
- v Unixech změny okamžitě viditelné
- immutable-shared-file – pokud je soubor sdílený nejde jej měnit (jednoduchá implementace)

Práva přístupu

- ACL (access control list)
  - seznam uživatelů a jejich přístupových práv
  - udržovat seznam může být netriviální (možnost doplnit role)
  - Denied ACL
- UNIX
  - ,,ACL" – pro vlastníka, skupinu, zbytek

Opravnění na platformě Windows

Základní skupiny oprávnění

oprávnění	soubor	adresář
read	čtení souboru	čtení souborů a podadresářů
write	zápis do souboru	přidání souboru nebo podadresářě
read and execute	čtení a spouštění souborů	čtení a spouštění souborů v adresáři
list folder contents	N/A	vylistování adresáře
modify	jako write + smazání	jako write + smazání adresáře a souborů v něm
full control	veškerá oprávnění	veškerá oprávnění

- ve skutečnosti jen logické pojmenování pro skupinu přesnějších oprávnění
- např. Traverse Folder/Execute File, List Folder/Read Data, Read Attributes, Read Extended Attributes, Create Files/Write Data, Create Folders/Append Data, Write Attributes, ...

Opravnění na platformě unix

- ,,ACL" – pro vlastníka, skupinu, zbytek
- novější unixy i ACL

oprávnění	soubor	adresář
read	čtení souboru	čtení obsahu adresáře
write	zápis do souboru	změna obsahu adresáře
execute	spuštění souboru	vstup do adresáře (včetně zpracování cesty)

Další příznaky

- setuid a setgid u souboru – spuštění souboru s právy vlastníka nebo skupiny
- setgid u adresáře – vytvořeny soubor v adresáři zdědí skupinu
- sticky bit u adresáře – pouze vlastník souboru, adresáře nebo root může přejmenovat nebo smazat soubor v daném adresáři (typicky `/tmp`)

Implementace souborových systémů: očekávané vlasnosti

- (budeme předpokládat souborové systémy pro práci s disky)
- schopnost pracovat se soubory a disky adekvátní velikosti
- efektivní práce s místem (evidence volného, nízká fragmentace)
- rychlý přístup k datům
- eliminace roztroušení dat na disku
- odolnost proti poškození při pádu systému (výpadku napájení) $\Longrightarrow$ rychlé zotavení

Další vlastnosti

- snapshoty
- komprese dat
- možnost zvětšovat/zmenšovat FS za běhu
- kontrolní součty
- defragmentace za běhu
- správa oprávnění
- atd.

Struktura disku

- pro jednoduchost předpokládáme, že struktura disku je lineární
- MBR – master boot record: informace o rozdělení disku na svazky + zavaděč
- nověji GPT (GUID Partition Table)
- Tan. 400
- *sektor disku* – obvykle velikost 4 kB, 512 B
- $\Longrightarrow$ pracuje se s většími bloky 1-32 kB (často 4 kB)
- $\Longrightarrow$ optimální velikost bloku? (rychlost vs. úspora místa)
- jednotlivé svazky obsahují souborový systém (vlastní organizace dat)
- VFS – virtuální souborový systém (Sta. 567)
- je potřeba si udržovat informace o jednotlivých souborech (FCB, inode)
- cache

Alokace diskového prostoru

Alokace souvislých bloků

- soubory jsou ukládany na disk za sebe v souvislých blocích
- rychlé sekvenční čtení, problém s uvolněnými soubory
- CD?

Soubor jako spojový seznam

- FS je rozdělený na bloky
- každý blok ma ukazatel na následující
- rychlé sekvenční čtení; problematický náhodný přístup (nutnost číst vše) + poškození disku
- varianta: uchovávání odkazů na bloky ve spec. tabulce (FAT)

Indexová alokace

- soubor si nese informaci o svém uložení v blocích (struktura na začátku souboru)
- problém s velkými soubory $\Longrightarrow$ víceúrovňové tabulky

Evidence volného místa

- je potřeba udržovat informace o volném místě
- použití spojového seznamu volných bloků (možné použít volné bloky); jako u souborů
- vylepšení $\Longrightarrow$ rozsahy volných bloků (problém při fragmentaci)
- bitmapy – každý bit udává, jestli je daný blok volný (nutné místo – obvykle méně než promile kapacity)

Přidělování volného místa

- je žádoucí zapisovat data do souvislých volných bloků $\Longrightarrow$ eliminace přesunů hlavičky
- heuristické algoritmy (složité na testování)
- problematické zaplnění disku nad 95 % a současné zapisování více velkých souborů

Adresáře

- organizace adresářů $\Longrightarrow$ vliv na výkon
- různé struktury
  - spojový seznam – jednoduchá implementace; větší složitost
  - hash tabulka (komplikace s implementací)
  - varianty B-stromů (časté u moderních FS)
- umístění informací o souborech
  - součást adresáře
  - součást souboru (UNIX) $\Longrightarrow$ problém: listování adresáře $\times$ možnost mít soubor ve více nebo žádném adresáři

Cache a selhání systému

- kvůli rychlejšímu přístupu nejsou často data zapisována přímo na disk $\Longrightarrow$ nejdřív do cache
- při výpadku (pád systému, výpadek napájení) nemusí být data ve write-back cache zapsána $\Longrightarrow$ poškození FS
- potřeba opravit FS (`fsck`, `chkdsk`) $\Longrightarrow$ časově náročné
- případné narušení systému
  - jeden uživatel zapíše data na disk a smaže je
  - druhý uživatel vytvoří velký soubor a potom zapsaní metadat vyvolá výpadek
  - po restartu čte data prvního uživatele

Řešení

- synchronní zápis $\Longrightarrow$ zpomalení, konzistence nemusí být zaručena
- soft updates – uspořádání zápisů podle určitých pravidel (*BSD)
- žurnálování
- zápis metodou copy-on-write

Žurnálování

- data se zapisují v transakcích (přesun FS z jednoho konzistentního stavu do druhého)

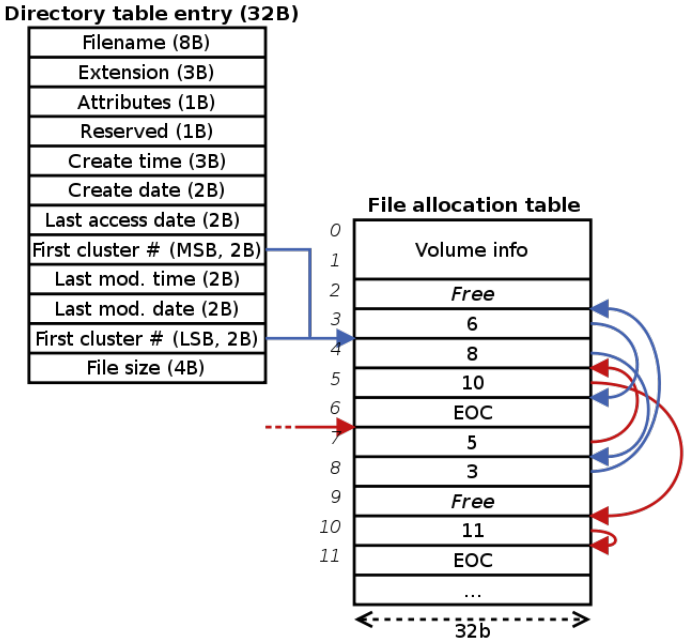
Algoritmus

1. nejdřív se změna (transakce) zapíše do žurnálu (logu)
2. po zapsání do žurnálu je **záznam označen spec. značkou**
3. data se zapíší na cílové místo na disk
4. po zapsání na disk je záznam z žurnálu odstraněn
5. při připojení FS se kontroluje stav žurnálu
   - zápis záznamu do žurnálu nebyl dokončen (chybí značka) $\Longrightarrow$ transakce se neprovede
   - jinak: transakce se zopakuje podle informací ze žurnálu

- často se žurnálují jen metadata
- žurnál je cyklický; při zaplnění se zapíší/uvolní ty na začátku
- je potřeba atomických zápisů na disk
- cache & buffery komplikují implementaci

FAT

- souborový systém pro MSDOS (přežil se až do Windows ME)
- jednoduchý design
- soubory se jmény ve tvaru 8.3, nepodporuje oprávnění
- nemá metody proti poškození dat
- disk rozdělený na bloky (clustery)
- soubory popsány pomocí File Allocation Table (FAT) – spojový seznam
- disk rozdělen na úseky:
  - bootsector (rezervovaná oblast) + informace o svazku
  - 2× FAT
  - kořenový adresář
  - data
- adresáře jako soubory; kořenový adresář je vytvořen při inicializaci
- původní FAT nepodporoval adresáře


Directory table entry (32B)
Filename (8B)
Extension (3B)
Attributes (1B)
Reserved (1B)
Create time (3B)
Create date (2B)
Last access date (2B)
First cluster # (MSB, 2B)
Last mod. time (2B)
Last mod. date (2B)
First cluster # (LSB, 2B)
File size (4B)
File allocation table
0
1
2
3
4
5
6
7
8
9
10
11
Volume info
Free
6
8
10
EOC
5
3
Free
11
EOC
...
32b

FAT: varianty

- FAT12, 16, 32: podle velikosti clusteru; (max. kapacity – 32 MB, 2 GB, 8 TB)
- další omezení na velikost souboru

Virtual FAT

- podpora dlouhých jmen (LFN)
- až 256 znaků
- soubor má dvě jména – dlouhé a ve tvaru 8.3
- dlouhá jména uložena jako další záznamy v adresáři

exFAT

- určen pro flash paměti
- podpora větších disků (512 TB/64 ZB)
- podpora v novějších Windows (povedně Windows CE 6)
- zatížen patenty

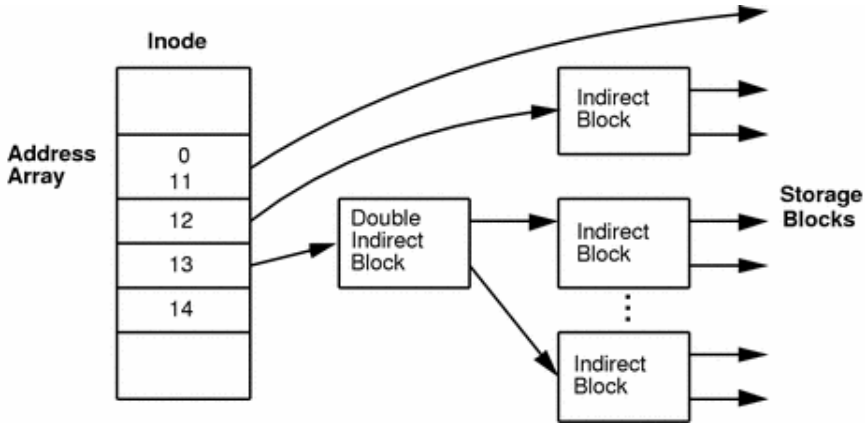
UFS: Unix File System (1/2)

- v různých variantách přítomný v unixových OS – *BSD, Solaris, System V, Linux (ext[234])
- disk se skládá:
  - bootblock – místo pro zavaděč OS
  - superblock – informace o souborovém systému
  - místo pro inody
  - místo pro data

Inoda

- struktura popisující soubor
- informace o souboru
  - typ souboru, vlastníka (UID, GID), oprávnění (rwx)
  - časy (vytvoření, přístup)
  - počet ukazatelů, počet otevřených popisovačů
- informace o uložení dat
  - patnáct ukazatelů na bloky na disku
  - bloky 0-11 ukazují na bloky dat
  - blok 12 – nepřímý blok 1. úrovně
  - blok 13 – nepřímý blok 2. úrovně
  - blok 14 – nepřímý blok 3. úrovně

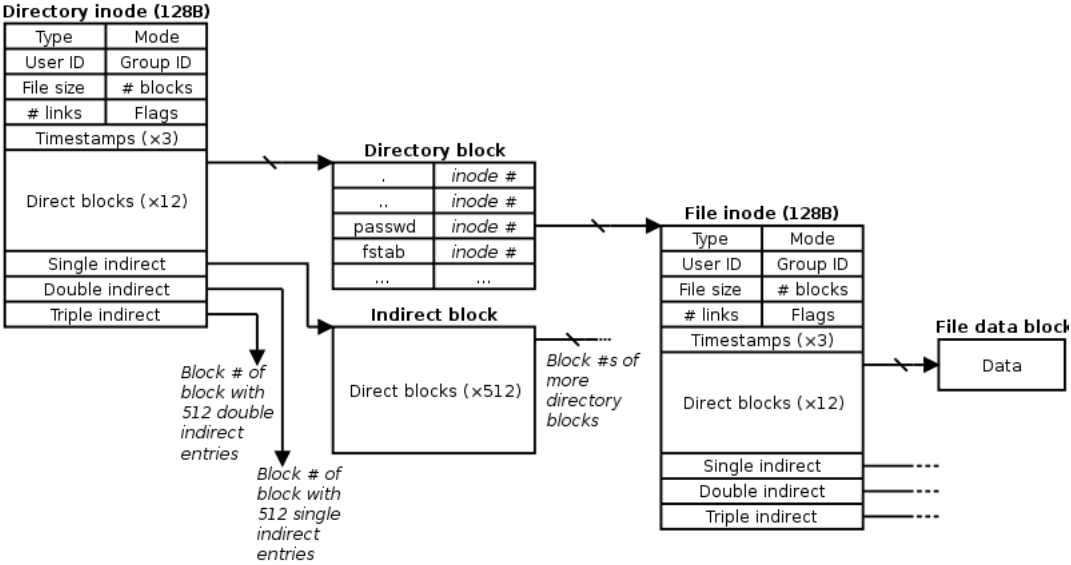
Inode


Inode
Address Array
0
11
12
13
14
Double Indirect Block
Indirect Block
Indirect Block
Indirect Block
Storage Blocks

UFS: Unix File System (2/2)

- struktura inode umožňuje mít *řídké soubory*
- adresář je soubor obsahující sekvenci dvojic (jméno souboru, číslo inode)
- k evidenci volného místa a inod se používají bitmapy
- svazek může být rozdělný na několik tzv. skupin – každá mající vlástní inode, bitmapy, atd. + kopie superblocku $\Longrightarrow$ sloučení souvisejících dat $\Longrightarrow$ eliminace přesunů hlavičky
- velikost bloku $\Longrightarrow$ rychlejší přístup k větším souborům vs. nevyužité místo
- možnost rozdělit blok na několik fragmentů
- konkrétní detaily se mohou lišit
- např. FreeBSD přidává možnost dělat snapshoty

Inode


Directory inode (128B)
Type
Mode
User ID
Group ID
File size
blocks
links
Flags
Timestamps (×3)
Direct blocks (×12)
Single indirect
Double indirect
Triple indirect
Directory block
.
inode #
..
inode #
passwd
inode #
fstab
inode #
...
...
Indirect block
Direct blocks (×512)
Block #s of more directory blocks
Block # of block with 512 double indirect entries
Block # of block with 512 single indirect entries
File inode (128B)
Type
Mode
User ID
Group ID
File size
blocks
links
Flags
Timestamps (×3)
Direct blocks (×12)
Single indirect
Double indirect
Triple indirect
File data block
Data

FS v Linuxu

- Linux nemá jeden hlavní FS
- nejčastěji se používá: Ext2/3/4
- název souboru může mít až 256 znaků (s výjimkou znaků / a \0)
- vychází z UFS
- ext2: maximální velikost souboru 16 GB–2 TB, disku: 2 TB – 16 TB
- ext3: přidává žurnál (3 úrovně – journal, ordered, unordered), binárně kompatibilní s ext2
- ext4: přidává vylepšení
  - maximální velikost souboru 16 TB–2 TB, disku: 1 EB
  - podpora extentů (místo mapování bloků je možné alokovat blok až do velikosti 128 MB)
  - optimalizace alokací
  - lepší práce s časem
- další FS: ReiserFS, BtrFS, JFS, XFS, ...

NTFS: Úvod

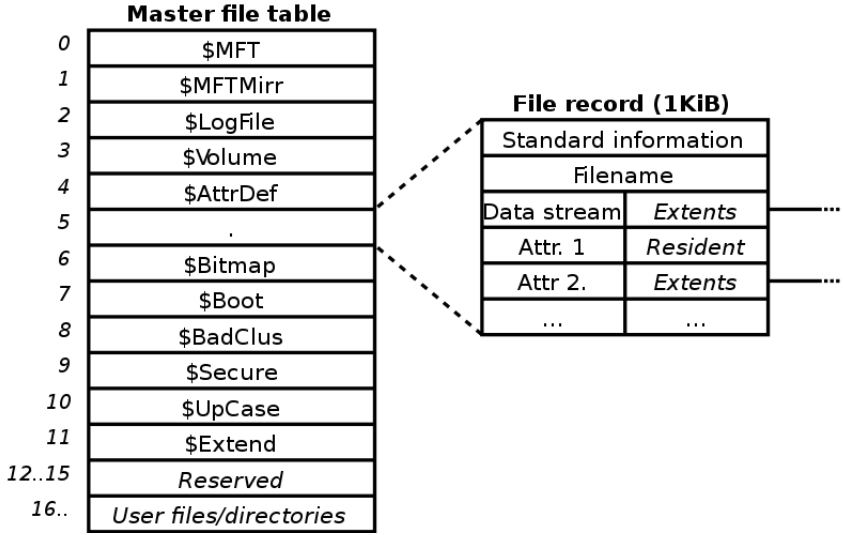
- hlavní souborový systém Windows NT
- kořeny v OS/2 a jeho HPFS (vyvíjen od roku 1993)
- velikost clusteru podle velikosti svazku (512 B–4 KB) $\Longrightarrow$ max. velikost disku 256 TB
  max. velikost souboru 16 TB
- oproti FAT (souborovému systému W9x) ochrana před poškozením + práva
- žurnálování a transakce
- podpora více streamů v jednom souboru
- dlouhé názvy (255 znaků) + unicode
- podpora standardu POSIX; hardlinky, symlinky
- komprese a řídké soubory

Adresáře

- opět technicky soubory; jména v B+ stromech
- některá metadata souborů jsou součástí adresáře

NTFS: Struktura disku (1/2)

- na začátku disku: boot sector
- 12 % MFT (Master File Table); 88 % data souborů
- MFT je soubor popisující všechny soubory na FS (MFT je taky soubor)
- MFT se skládá ze záznamů o velikosti 1 KB
- každý soubor je popsán tímto záznamem
- 32 prvních souborů má speciální určení ($MFT, $MFTMirr, $LogFile, $Volume, $Bitmap, $Boot, $BadClus, ...)
- informace o souborech včetně jména, časů, atd. uloženy jako záznam v MFT jako dvojice *atribut-hodnota*
- tělo souboru je taky atribut $\Longrightarrow$ uniformní přístup; možnost uložit malé soubory přímo do MFT
- alternativní proudy $\Longrightarrow$ opět atributy
- v případě potřeby může jeden soubor zabrat víc záznamů v MFT
- případně lze použít místo mimo MFT (rezidentní a nerezidentní atributy)


Master file table
0
$MFT
1
$MFTMirr
2
$LogFile
3
$Volume
4
$AttrDef
5
.
6
$Bitmap
7
$Boot
8
$BadClus
9
$Secure
10
$UpCase
11
$Extend
12..15
Reserved
16..
User files/directories
File record (1KiB)
Standard information
Filename
Data stream
Extents
Attr. 1
Resident
Attr 2.
Extents
...
...

NTFS: Struktura disku (2/2)

- data v souboru jsou popsána pomocí (atributu) tabulky mapující VCN (virtual cluster number) na LCN (logical cluster number)
- VCN – číslo clusteru v souboru (indexováno od nuly)
- LCN – číslo clusteru ve svazku
- každý záznam v tabulce je ve tvaru: VCN, LCN, počet clusterů, např.

VCN	LCN	počet
0	42	4
4	123	8
32	456	15

Komprese

- řídké soubory
- možnost transparentně komprimovat obsah (vždy po 16 clusterech) $\Longrightarrow$ bloky dat zarovnány na 16 clusterů; pokud zabírá míň místa, je komprimován
- čtení i zápis provádí (de)kompresi (LZ77) $\Longrightarrow$ dopad na výkon

ISO-9660

- souborový systém pro CD-ROM; podpora všech OS
- zápis jen jednou; sekvenční čtení $\Longrightarrow$ není potřeba dělat kompromisy
- logický sektor 2048 B (může být i větší)
- na disku může být víc logických svazků; svazek může být na více discích
- na začátku 16 rezervovaných bloků + 1 blok (Primary Volume Descriptor) $\Longrightarrow$ informace o disku; odkaz na kořenový adresář
- adresář popsán pomocí záznamů proměnlivé délky (viz Tan. 432)
  - textová data v ASCII
  - binární 2× (little- i big-endian)
- možnosti formátu určeny úrovněmi a rozšířeními
- **Level 1** – soubory 8.3; všechny soubory spojité; 8 úrovní adresářů
- **Level 2** – jména až 31 znaků
- **Level 3** – nespojité soubory (jednotlivé souvislé bloky se mohou opakovat)

ISO-9660: Rozšíření

Rock Ridge

- kompatibilita s unixy
- přidává dlouhá jména
- neomezené zanoření adresářů
- unixová oprávnění
- podpora symbolických odkazů; možnost mít na disku soubory zařízení

Joliet

- kompatibilita s Windows
- přidává dlouhá jména + podporu Unicode
- neomezené zanoření adresářů

UDF: Universal Disk Format

- náhrada za ISO-9660
- používán převážně pro DVD a Blue-ray disky
- dlouhé názvy, soubory až 1EB
- různé varianty formátu:
  - **Plain build** – základní formát (data lze přepisovat pokud médium podporuje; přepis konkrétních sektorů – DVD-RAM, DVD+RW)
  - **Virtual Allocation Table** – inkrementální zápisy (CD-R)
  - **Spare build** – pokud to médium podporuje, lze data přepisovat; zahrnuta obrana proti opotřebení sektorů