KIV Operační systémy

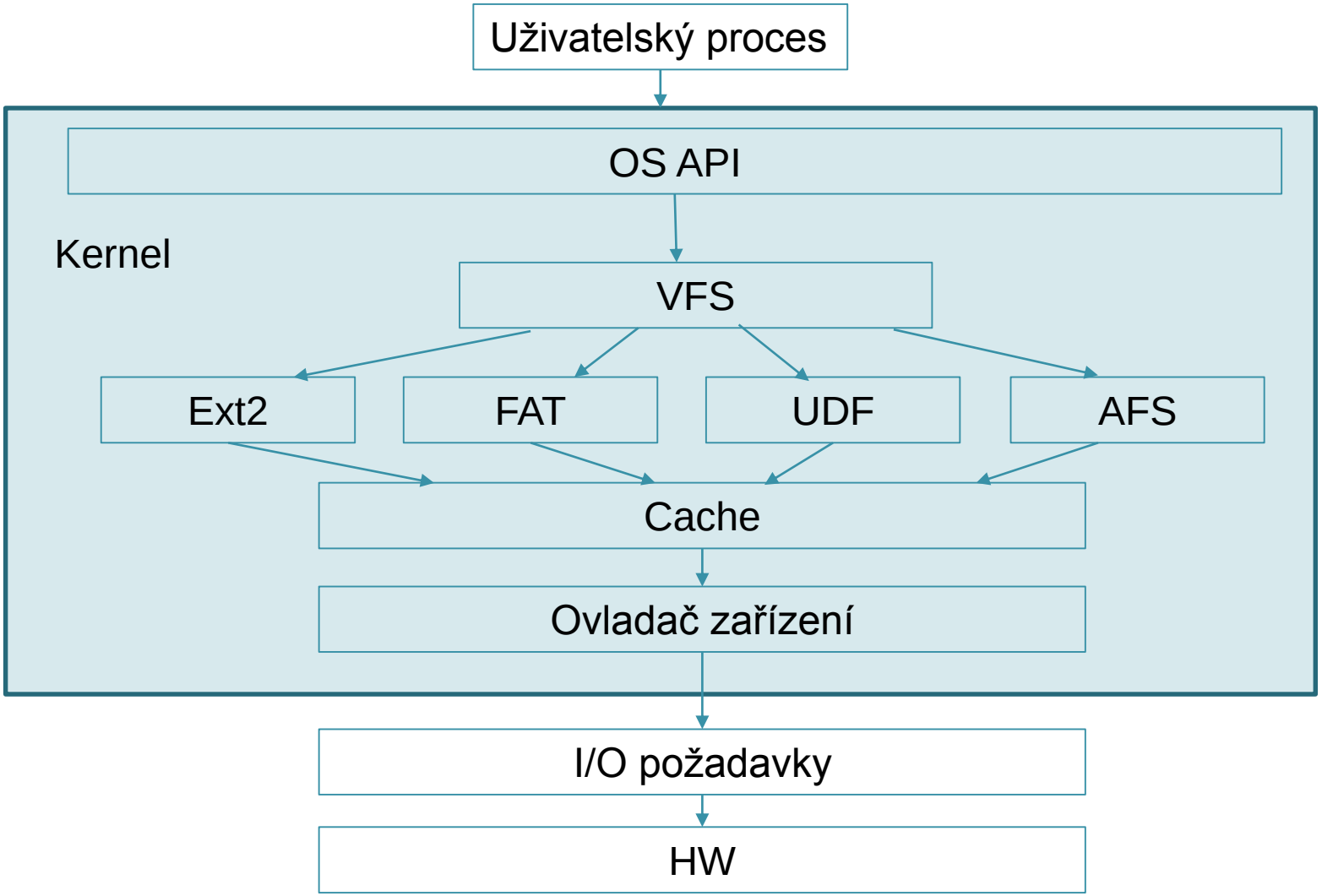
Souborový systém

Verze 1.0

Virtual File System

- Thread volá jednotné API pro práci se soubory
  - Adresář je jenom speciální typ souboru
- Souborový systém je abstrakce nějakého souvislého bloku paměti, které ho umožňuje organizovat do souborů – tj. do menších, pojmenovaných bloků paměti
- Typicky je blok paměti hw realizovaný diskem, ale může to být i síťový protokol nebo část RAM
- Každý blok paměti může mít jiný souborový systém
- OS musí zpřístupnit jednotné API – Virtual File System

Virtual File System



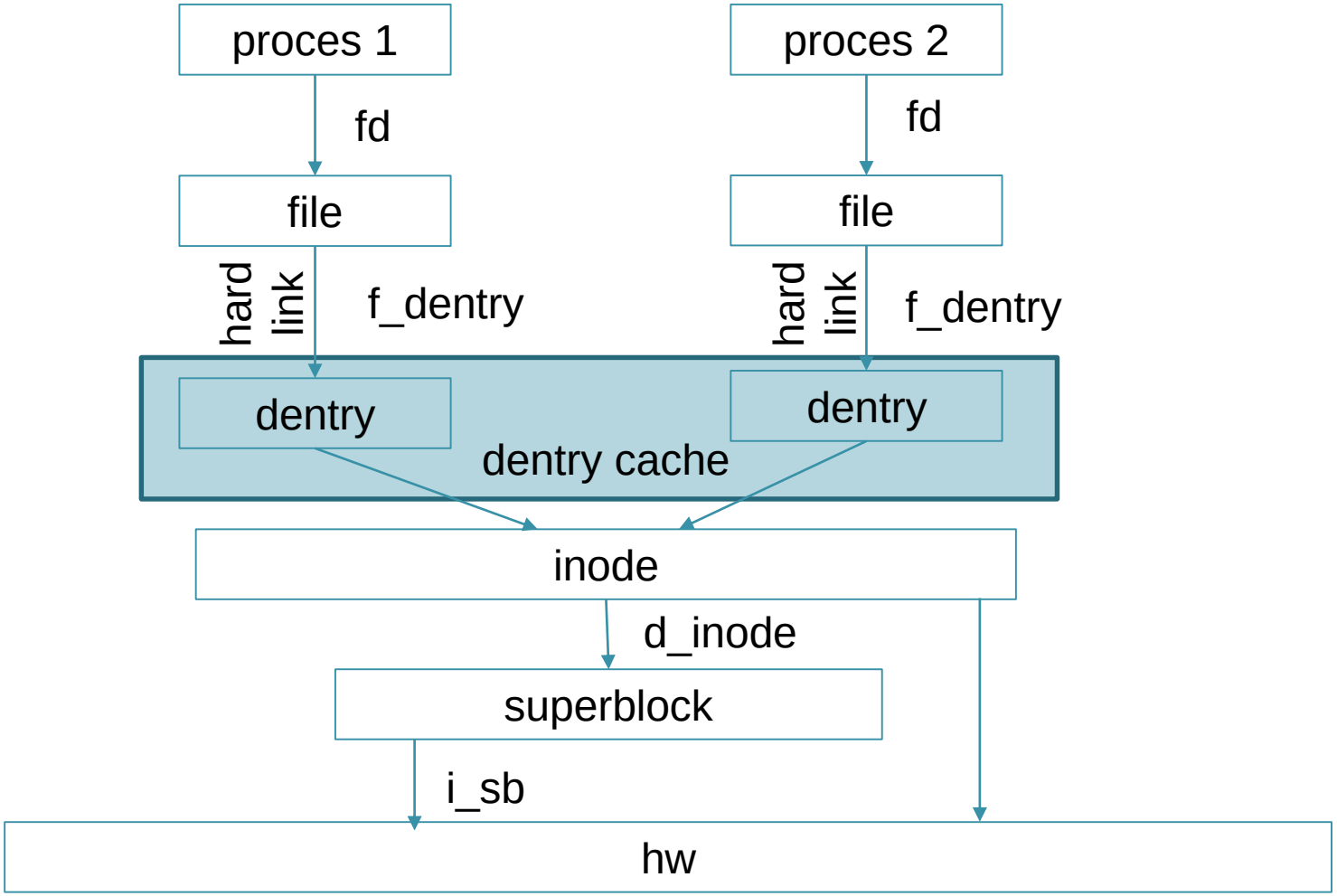
VFS koncept

- VFS je sice obecný model souborového systému, ale má design podle UNIXu
  - Souborové systémy, které byly navržené podle jiných pravidel, se mu musí přizpůsobit – např. FAT nemá koncept inode
- Hlavní komponenty VFS:
  - superblock – info o připojeném souborovém systému
  - inode – info o konkrétním souboru
  - file – info o konkrétním, otevřeném souboru
  - Dentry – info o adresářové položce

VFS koncept

- VFS je sice obecný model souborového systému, ale má design podle UNIXu
  - Souborové systémy, které byly navržené podle jiných pravidel, se mu musí přizpůsobit – např. FAT nemá koncept inode
- Hlavní komponenty VFS:
  - superblock – info o připojeném souborovém systému
  - inode – info o konkrétním souboru
  - file – info o konkrétním, otevřeném souboru
  - Dentry – info o adresářové položce

Virtual File System



VFS tabulka funkcí

- V OOP bychom nadefinovali abstraktní třídy jednotlivých objektů VFS, a konkrétní implementace souborových systémů by je implementovaly
- Linux má C rozhraní, a v C není OOP
- =>každý VFS má definovanou sadu funkcí, které nad ním lze provádět a instance každého objektu je reprezentována tabulkou ukazatelů na funkce
  - Obdoba VMT
  - Některé generické funkce může poskytnout už OS

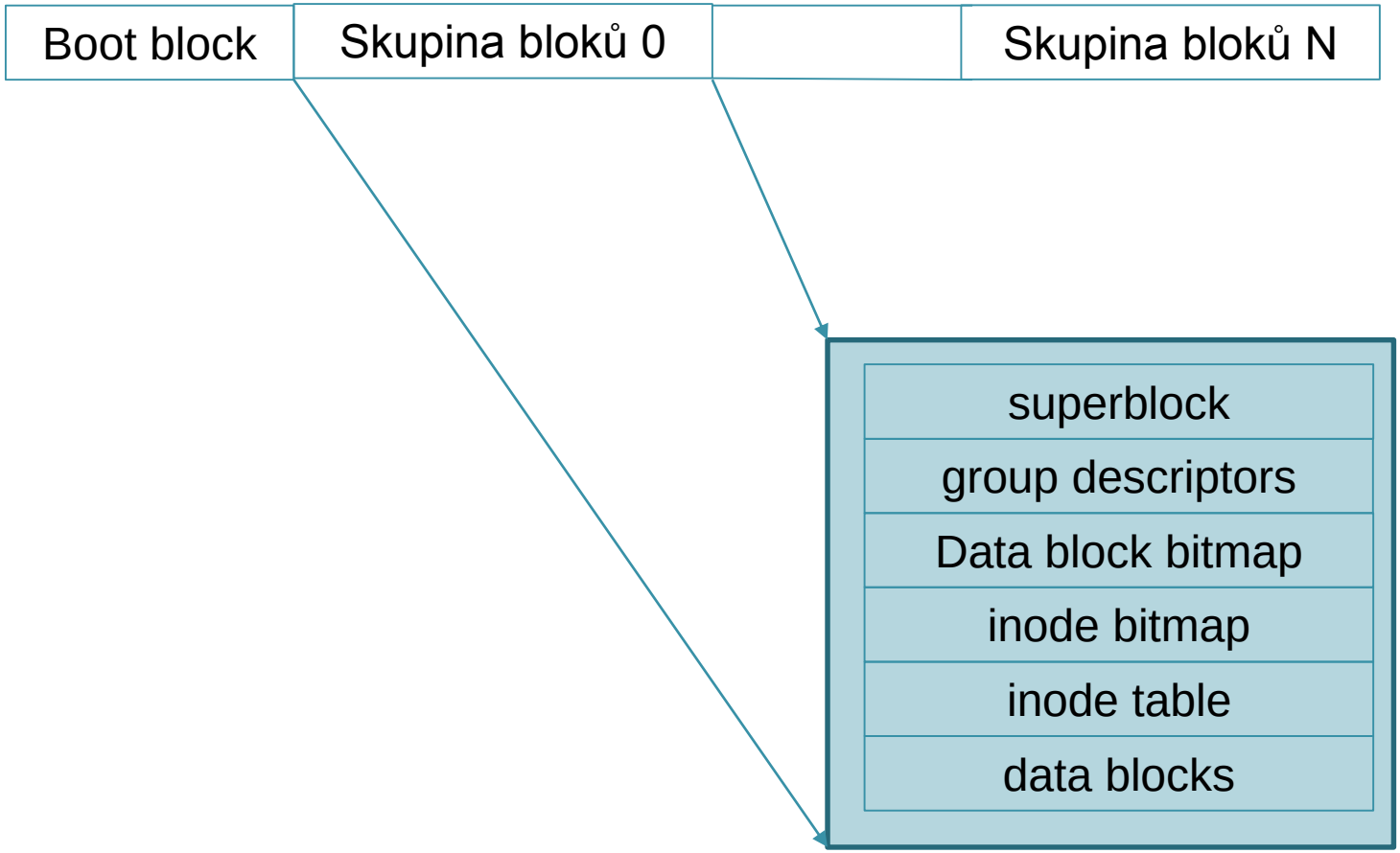
Proces a soubory

- Každý proces má svůj pracovní a kořenový adresář
  - Uloženo ve *fs_struct* ve *fs* položce PCB
- Otevřené soubory jsou uloženy ve *files_struct* v položce *files* PCB
  - Funkce otevření souboru open() vrací index do pole objektů *file* (položka *fd* ve *files*)
  - current->files->fd[0] je stdin, 1 stdout a 2 stderr
  - Po uzavření close() nějakého fd, open() vrátí první volný index do fd – tj. lze takto přesměrovat stdio

Extended File System 2

- ext1 … ext4, klíčový pro pochopení je ext2
- Návrhovým vzorem odpovídá VFS
- Skládá se z bloků, které seskupuje
  - Velikost bloku je od 1 do 4kB
  - Volitelný počet inode
  - Prealokuje bloky souborům před tím, než jsou použitý
  - První blok je vždy vyhrazen pro boot sector

Struktura ext2



- První skupina bloků, tj. 0, je vyhrazena pro jádro OS

superblock

- Obsahuje
  - Celkový počet inode uzlů
  - Velikost souborového systému v blocích
  - Počítadla volných bloků a inode uzlů
  - Velikost bloku
  - Počty bloků a inodů uzlů ve skupině
  - 128-bit id souborového systému
  - Počítadlo připojení
  - A další

group descriptor

- ext2_group_desc Obsahuje
  - Počty bloků bitmap bloků, inode uzlů a prvního inode v tabulce bloků
    - Pokud je n-tý bit bitmapy nastaven, daný blok/inode je použit
  - Počty volných bloků, inode uzlů a adresářů v bloku
  - A další

Tabulka inode uzlů

- Jsou to po sobě uložené bloky obsahující záznamy typu ext2_inode, tj. o stejné velikosti
- ext2_inode obsahuje
  - Typ souboru a přístupová práva
  - Identifikátory vlastníka a skupiny
  - Délku v bytech a blocích
  - Časová razítka
  - Pole ukazatelů na datové bloky
  - A další

Typy souborů

- inode.filetype
  - Běžný soubor – potřebuje datové bloky, kde ukládá data
  - Adresář – speciální typ souboru, jeho datové bloky ukládají jména souborů v adresáři společně s jejich čísly inode uzlů
  - Symbolický odkaz – do 60 bytů se odkaz ukládá v inode (tzv. fast symbolic link), jinak také potřebuje datové bloky

ext2 paměťové struktury

- Časté operace nad souborovým systémem vyžadují frekventovaný přístup k některým datovým strukturám
  - => lze je při připojení systému nahrát do paměti, aktualizovat je tam a průběžně a při odpojení je nahrát na disk
- OS sice drží část disku v diskové cache, ale proč tam zabírat místo datům něčím, co stejně potřebujeme v mít paměti?

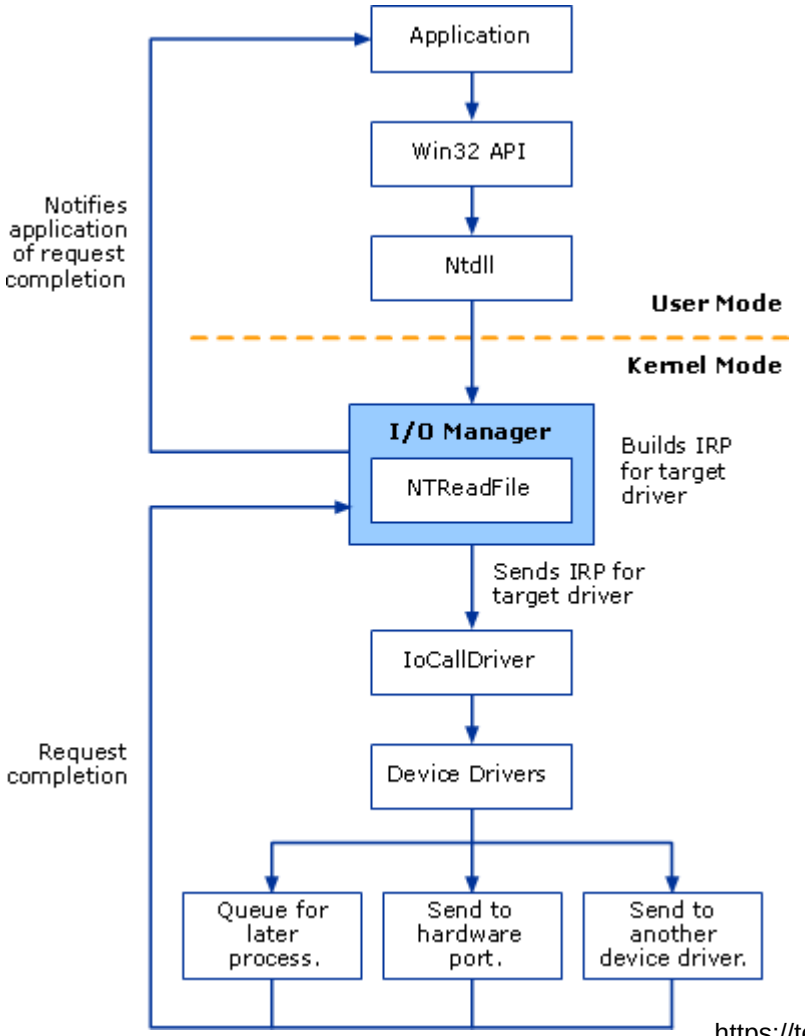
Installable File System

- VFS Windows
- Funguje ve třech režimech
  - nemusí být nutné implementovat všechny tři
  - file system – vytváří vlastní souborový systém na diskem
  - Mini Filter – rozhraní pro antivirové programy a indexovací služby
  - FS FilterDriver – používá se pro úpravu již existujících souborových systémů
    - Zachycuje požadavky a vrací modifikované odpovědi
- Používá IO Request Packet pro komunikaci

IO Request Packet

- Struktura používaná ke komunikaci mezi ovladačem zařízení a OS
- Popisuje požadavky, které se mají se provést
- Dávají se do fronty, kterou si OS může přeuspořádat
- Většinou je vytváří I/O manager podle volání souborových funkcí z uživatelského adresového prostoru
- Ale mohou je vytvářet i další části, např. systém úspory energie bude chtít při nečinnost vypnout disk

IO Request Packet

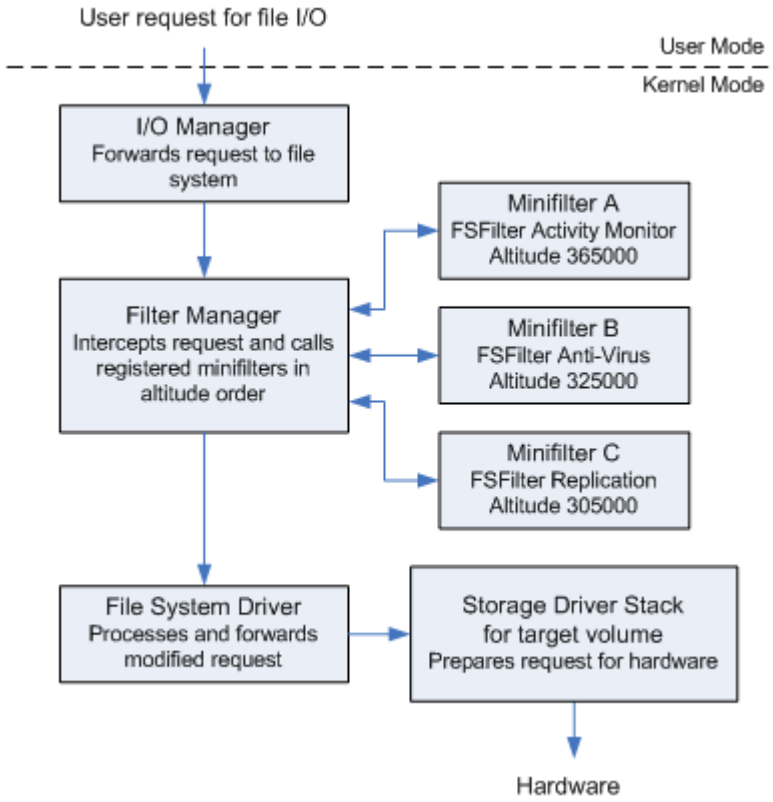


https://technet.microsoft.com/en-us/library/cc776371%28WS.10%29.aspx

Filter Manager

- Aktivuje se při načtení mini filtru
- Připojí se do file system stacku daného disku
  - Respektive zaregistruje se pro I/O operace, které bude chtít filtrovat
- Filtry jsou ve stacku podle priorit, např. antivir má větší prioritu než indexovací či replikovací služba

Filter Manager



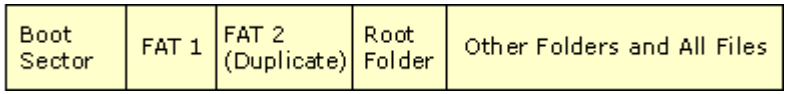
https://msdn.microsoft.com/en-us/library/windows/hardware/ff541610%28v=vs.85%29.aspx

Fast I/O

- IRP je výchozí mechanismus pro všechno
  - Synchronní i asynchronní přenosy, data v cache i mimo ni, výpadky stránek
- Ale pro synchronní operace nad daty v cache lze použít Fast I/O
- Data jsou pak přenesena přímo mezi buffery procesů přes systémovou cache
  - Zkratka, která obejde celý fs a hw stack – něco jako loopback na localhostu
- Fast I/O dělá v případě neúspěchu fallback na IRP

File Allocation Table

- První verze byla nasazena roku 1977 na 8 palcových disketách, v dalších verzích se používá dodnes
  - Digitální kamery, bootování UEFI, USB čitelné různými OS…
- Disk s FAT má bootovací sektor, alokační tabulku souborů, její kopii, kořenový adresář a zbývající adresáře a soubory



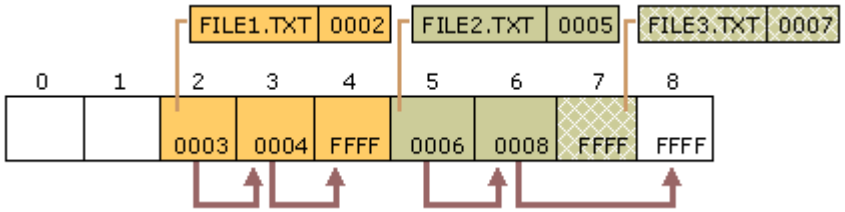
https://technet.microsoft.com/en-us/library/cc938438.aspx

FAT – položka adresáře

- Obsahuje
  - Jméno ve formátu 8.3
    - Dlouhá jména řeší VFAT a pozdější verze
  - Atributy
    - Disk, adresář, soubor
      - Skrytý, systémový, jen ke čtení
  - Časová razítka
  - Číslo prvního clusteru, kde začínají data položky adresáře
  - Velikost souboru
  - A další

FAT alokace souborů

- Souborům se při zápisu přiděluje první volný cluster
  - FAT disk je rozdělen na bloky nazývané clustery
  - Clustery nemusí jít po sobě => problém fragmentace
- Každý cluster obsahuje číslo dalšího cluster, který obsahuje další data daného souboru, anebo značku konce souboru



https://technet.microsoft.com/en-us/library/cc938438.aspx

New Technology File System

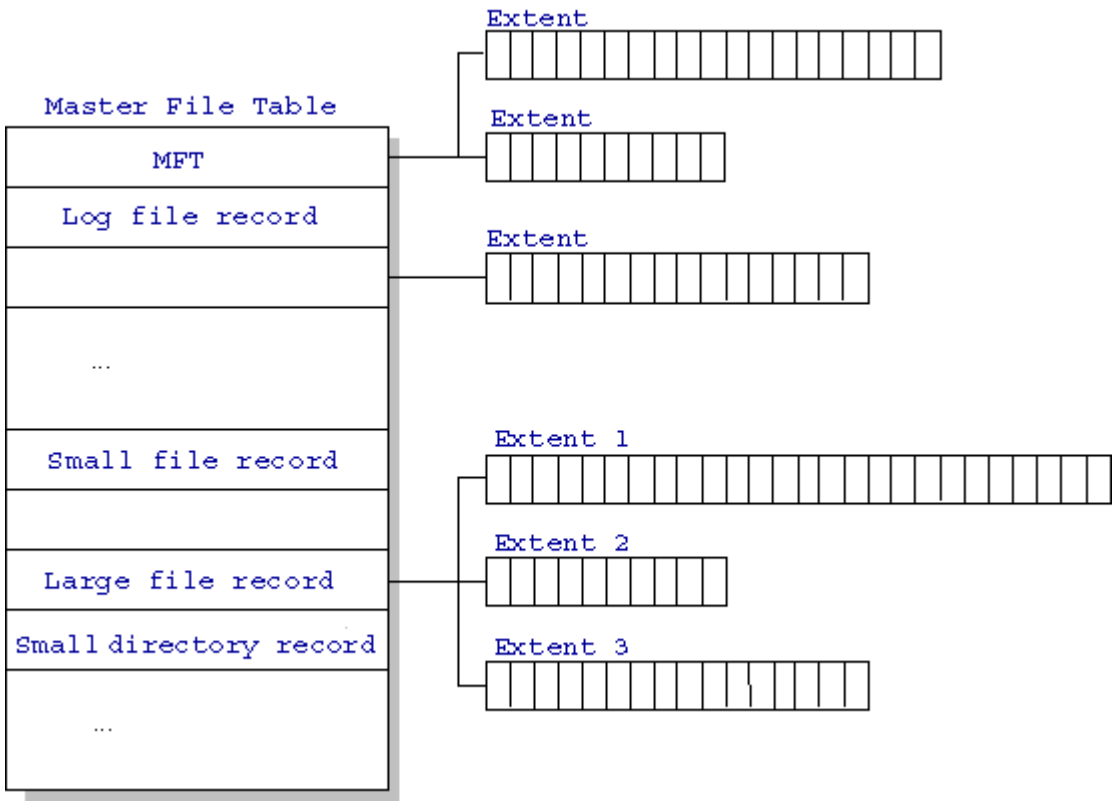
- Navržen pro Windows NT, aby na rozdíl od FAT a HPFS (fs vyvíjený s IBM pro OS/2) uměl
  - metadata, ACL, journaling, datové streamy, atd.
  - ale hlavně, aby se zvýšila rychlost, spolehlivost a zlepšilo se využití místa na disku
  - Např. od Vista umí i transakce
- Detailní srovnání např. NTFS, ext4, či btrfs je mimo rozsah a je ponecháno na laskavém čtenáři „as needed“

NTFS disk

- Disk naformátovaný s NTFS má následující strukturu:
  - Boot sektor
  - Master File Table
  - Master File Table zóna, do které může MFT růst
  - Systémové soubory
  - Uprostřed disku je kopie (části) MFT
  - Zbývající místo je určeno pro soubory

NTFS's MFT

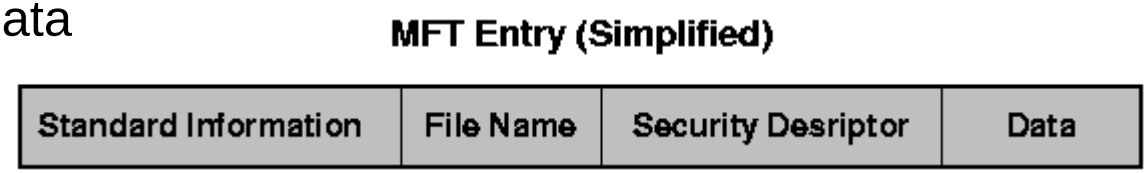
- Každá položka představuje jeden soubor



http://ntfs.com/ntfs-mft.htm

MFT položka

- Položka se sestává z dvojic <atribut, hodnota>, takže ji lze dynamicky rozšiřovat, a obsahuje
  - Standardní informace
    - Práva, časová razítka, hard-link count (kolik adresářů na soubor ukazuje)
  - Jméno souboru
  - Security descriptor
  - Data



http://pages.cs.wisc.edu/~bart/537/lecturenotes/s26.html

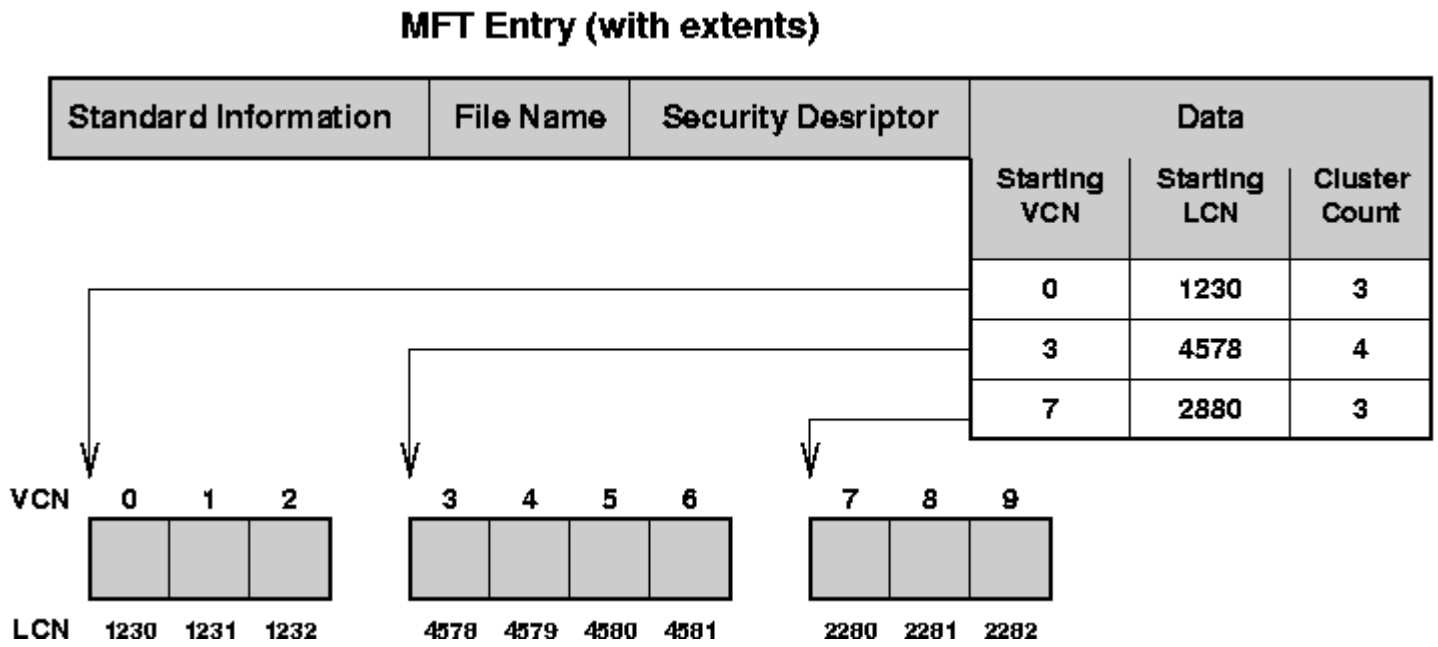
MFT položka souboru

- Každý soubor se skládá alespoň z jednoho data stream
  - type soubor.txt:druhytext
- Pokud jsou celá data souboru větší než místo v MFT, pak položka data odkazuje na další místo na disku, kde jsou data uložena
- Dostatečně malé souboru jsou uloženy přímo v MFT
  - Viz fast symbolic link u ext2

MFT data pointer

- Pointery na data jsou pointery na posloupnost logických clusterů na disku (extent)
- Každá posloupnost má
  - VCN – virtual cluster number, první cluster souboru
  - LCN – logical cluster number, první logický cluster jedné sekvence
  - Délka v počtu clusterů
- MFT obsahuje list položek popisujících extent
  - Unix-like používá strom

MFT data pointer

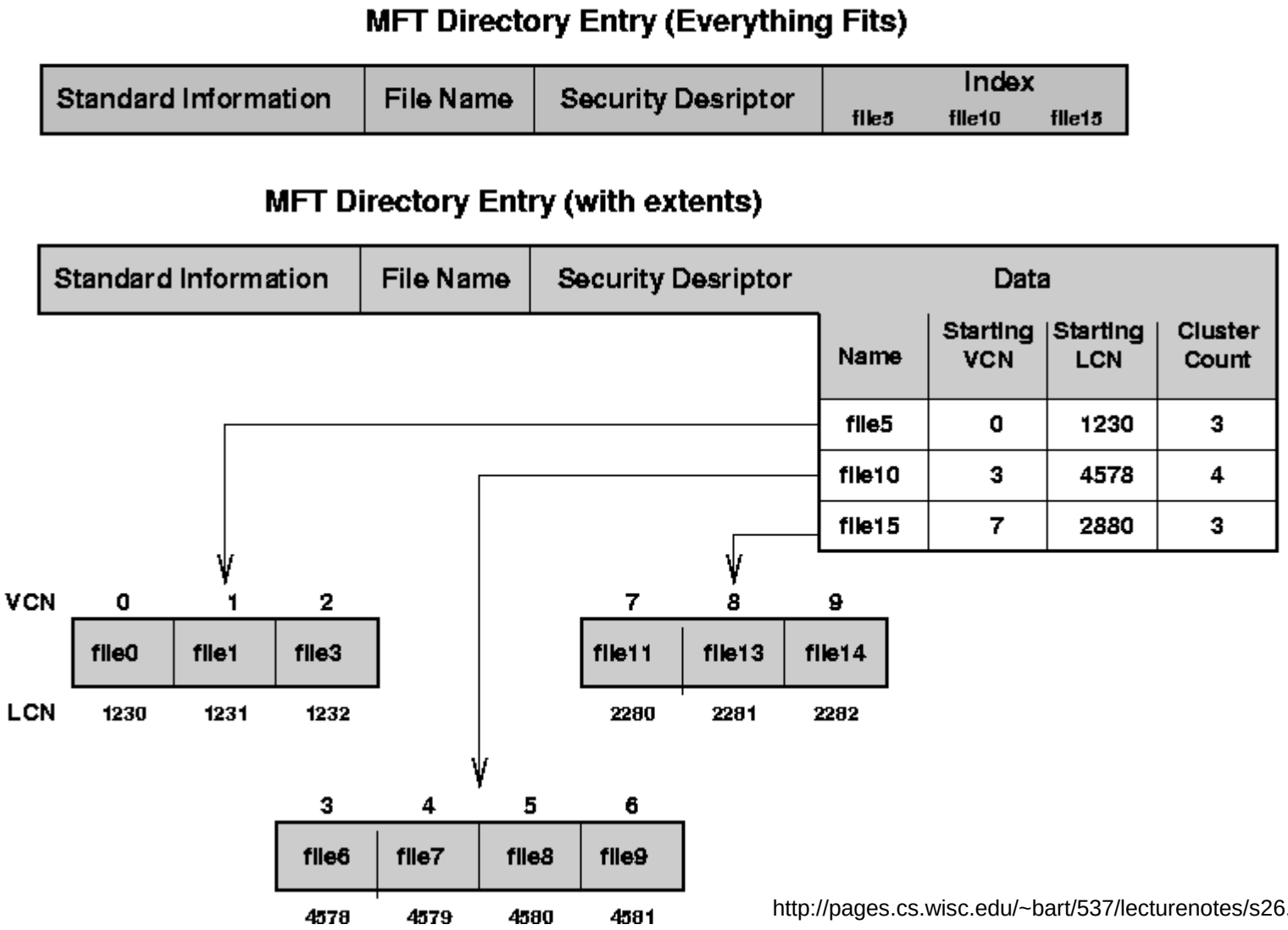


http://pages.cs.wisc.edu/~bart/537/lecturenotes/s26.html

MFT položka adresáře

- Adresář je speciální soubor obsahující seznam souborů
- Adresář má jméno a referenci
  - Reference je pár <číslo souboru, sekvenční číslo>
  - Číslo souboru je offset do MFT
    - Něco jako číslo inode uzlu ve VFS
- Položka adresáře obsahuje seznam souborů v B+ stromu
  - Jméno souboru je jak v položce adresáře, tak i v MFT
- Pokud je záznam adresáře dostatečně malý, je v MFT

MFT položka adresáře



http://pages.cs.wisc.edu/~bart/537/lecturenotes/s26.html