Operační systémy 2

Souborové systémy a jejich implementace

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Souborové systémy: Motivace

- potřeba uchovávat větší množství dat (primární paměť nemusí dostačovat)
- data musí být perzistentí (musí přežít ukončení procesu)
- k datům musí být umožněn souběžný přístup
- $\Longrightarrow$ řešení v podobě ukládání dat na vnější paměť (např. disk)
- $\Longrightarrow$ data ukládána do souborů tvořících souborový systém (File System/FS)
- soubor jako proud bytů (doprovázen doplňujícími informacemi)
- souborový systém jako abstrakce (odstínění od implementačních detailů)
- $\Longrightarrow$ Unix (dotaženo v Plan 9 from Bell Labs)
- zajímavý problém: pojmenovávání objektů (souborů) & jejich organizace

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 práci `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 `\n`, `\r\n`, `\r`)

Organizace souborů (1/3)

- soubory jsou rozlišované 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, i.e., 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
- speciální adresáře v každém adresáři „.“ a „..“ (aktuální adresář, nadřazený adresář) $\Longrightarrow$ nutné pro navigaci v hierarchii adresářů
- operace s adresáři: Create, Delete, OpenDir, CloseDir, ReadDir, Rename
- struktura nemusí být hierarchická $\Longrightarrow$ obecný graf (acyklický, cyklický)
  - hardlink – ukazatel na soubor (jeho tělo/obsah)
  - symlink – soubor obsahuje cestu (odkaz) k jinému souboru
- v Unixech běžné, ve Windows delší dobu (ale chybělo rozhraní)

Organizace svazků

- každý fyzický disk se skládá z jedné nebo více logických částí (partition; oddíl); popsané pomocí partition table daného disku
- v každém oddílu 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)
  - (jako důsledek lze snadno portovat jednotlivé FS)
- 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 (jiná řešení: více proudů bytů, sekvence záznamů, dvojice klíč-hodnota)
- sekvenční vs. náhodný přístup
- někdy může být výhodná jiná struktura
- rozdělení jednoho souboru na více proudů (např. multimediální soubor – video/audio stopy)
- 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 (jeden bit pro každý přístup)
  - udržovat seznam může být netriviální (možnost doplnit role)
  - Denied ACL

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 plnohodnotné 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)

Příklad

```
rwxrw-r-- 1 joe admin 4096 Nov 20 foo.txt
```

Další příznaky

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

Implementace souborových systémů: očekávané vlastnosti

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

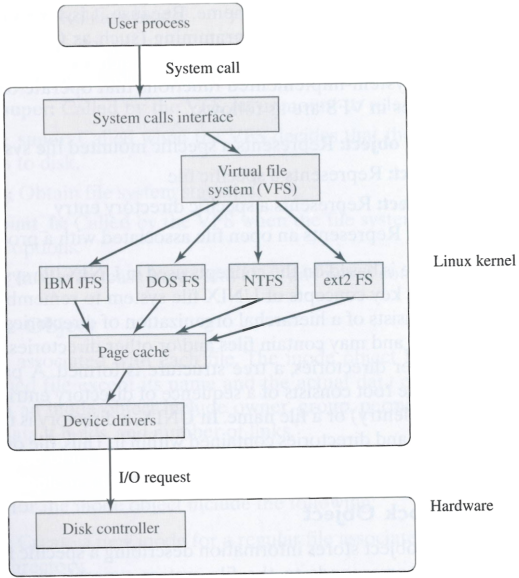
Další vlastnosti

- 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
- informace o rozdělení disku na oddíly + zavaděč (dříve: MBR – master boot record, nověji: GPT – GUID Partition Table)
- *sektor disku* – obvykle velikost 4 kB (starší disky 512 B)
- $\Longrightarrow$ pracuje se s většími bloky 1-32 kB (často 4 nebo 8 kB)
- $\Longrightarrow$ optimální velikost bloku? (rychlost vs. úspora místa)
- jednotlivé oddíly obsahují souborový systém (vlastní organizace dat)
- je potřeba si udržovat informace o jednotlivých souborech (FCB, inody), volné místo, apod.
- VFS – virtuální souborový systém
- cache

Virtuální souborový systém (VFS)



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 na 95 % + současné zapisování více velkých souborů
- např. algoritmus v ext2+
  - zvolí se cíl zápisu (první volný blok, příp. první volný blok za souborem)
  - hledá se v bitmapě první volných 8 bitů (8 bloků) ležících za cílem; pokud takové nejsou, hledá se po bitech
  - (prealokace) alokátor se snaží zabrat 8 bloků za sebou
- clusterování – souvislý zápis více bloků (možnost je přeskládat); FreeBSD

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 (metadat)
  - součást adresáře
  - součást souboru (UNIX) $\Longrightarrow$ problém: listování adresáře; jednoduché 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 po zapsání 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 (používá se i v databázích)
- COW (aktuální data se nepřepisují)

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**, indikující úplnost transakce
3. data se zapíší na cílové místo na disku
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