Organizace dat na linuxových souborových systémech

Petr Krčmář

ROOT.CZ_ vpsFree.cz

11. června 2022

O mně

- linuxák od roku 1998
- správce serverů
- lektor a konzultant
- šéfredaktor Root.cz
- člen vpsFree.cz
- organizátor LinuxDays
- můj web je petrkrcmar.cz

ROOT.CZ_ vpsFree.cz

Prezentace už teď na webu

https://www.petrkrcmar.cz

ROOT.CZ_ vpsFree.cz

Začneme rovnou ukázkou

- když neznáte věci pod povrchem
- překvapí vás, že se někdy chovají zvláštně
- divné chování pak vyplaší i letitého linuxáka
- proto malá ukázka *divného chování*

Ukázka: divné věci hned na úvod

Cože?

Souborový systém

- způsob organizace dat na blokovém zařízení
  - abstrakce určená pro uživatele
  - uživatel pracuje se soubory a adresáři
  - samotná organizace je před ním skrytá
- blokové zařízení je jednorozměrné úložiště
  - chybí jakákoliv struktura či členění
  - adresují se jednotlivé bloky
- souborový systém je stromová struktura s metadaty
  - adresáře, podadresáře, soubory
  - adresují se konkrétní objekty (soubory)
- budeme se věnovat souborovým systémům ext[2/3/4]

Základem je inode

- inode (index node, indexovaný uzel) je hlavní jednotkou
  - drží informace o každém objektu v souborovém systému
  - jeden inode = metadata o jednom objektu (souboru)
- každý inode má na jednom souborovém systému unikátní číslo
  - ID souborového systému + číslo inodu = konkrétní soubor
- inody jsou uloženy v **tabulce inodů** a jsou seřazeny podle čísla
  - vyhledání konkrétního je snadné, jeho pozici lze vypočítat
- tabulka vzniká se souborovým systémem
- když dojdou inody, není možné vytvářet další soubory
  - při obvyklých scénářích k tomu nedochází

Tabulka inodů

- je vytvářena při tvorbě souborového systému
- výchozí stav je jeden inode na 16 kB datového prostoru
  - výchozí hodnota je rozumná, inody vám nedojdou
  - příklad: na mém `/home` je obsazeno 60 % prostoru, ale jen 3 % inodů
- nemá smysl volit méně než velikost bloku (obvykle 4 kB)
  - měli bychom více inodů než bychom mohli využít
- číslo inodu je 32bitové, takže maximum je 4 miliardy

Zobrazení počtu inodů

```
# mkfs.ext4 /dev/sdb1
# df -i /dev/sdb1
```

ROOT.CZ_ vpsFree.cz

Ukázka: informace o inodech

Co obsahuje inode

- vlastní ID daného inodu
- typ objektu (soubor, adresář, odkaz…)
- informace o právech (rwx), ACL
- vlastník souboru a vlastnická skupina
- velikost souboru
- počet bloků obsazených daty
- počítadlo odkazů na inode
- časy vytvoření, přístupu a modifikace
- ukazatele na bloky s tělem souboru
- kontrolní součet metadat

Co obsahuje inode

- vlastní ID daného inodu
- typ objektu (soubor, adresář, odkaz…)
- informace o právech (rwx), ACL
- vlastník souboru a vlastnická skupina
- velikost souboru
- počet bloků obsazených daty
- počítadlo odkazů na inode
- časy vytvoření, přístupu a modifikace
- ukazatele na bloky s tělem souboru
- kontrolní součet metadat
- **Nechybí tu něco podstatného?**

Název souboru!

- názvy souborů opravdu **nejsou součástí inodu**
- názvy souborů jsou uloženy v adresářích
  - adresář je jen speciální případ souboru
  - v inodu je uložen *typ souboru* = tohle je adresář
  - v těle obsahuje seznam názvů souborů a jejich inodů
  - v tomto adresáři se nacházejí následující soubory…
  - souborový systém ví, že se k adresáři má chovat jinak
- název souboru nemá k objektu přímou vazbu
- v inodu je jen počítadlo výskytů v adresářích

ROOT.CZ_ vpsFree.cz

Hard linky

- jeden objekt může být **odkazován vícekrát**
- jeden soubor může mít více názvů – nevznikají nové objekty
- dokonce může ležet i ve více adresářích
  - tohle je v unixových systémech naprosto běžná věc
- na objekty se odkazuje **pomocí čísel inodů**
- odkazy na adresáře jsou zakázány kvůli nebezpečí vzniku smyček

Zobrazení inodů v adresáři

```
# ls -ia
# stat soubor.txt
# ln soubor.txt novy.txt
```

ROOT.CZ_ vpsFree.cz

Ukázka: vytvoření hard linku

Proti tomu soft linky

- soft nebo symbolic link je soubor obsahující odkaz
- vznikne **úplně nový objekt** s jiným inodem
- opět je v inode příznak = tohle je odkaz
- v těle je uložena cesta – relativní nebo absolutní
- souborový systém cestu automaticky interpretuje
  - pracuje se s cílem (target)
- pokud se cíl změní, odkaz na něj se **neupraví**
  - link se tak může rozbít (broken link)
  - nebo může ukazovat na jiný objekt

Vytvoření soft linku

```
# ln -s soubor.txt novy.txt
```

Ukázka: vytvoření soft linku

Smazání souboru

- soubor se maže **vždy v adresáři**
- odstraní se tím vlastně jen ze seznamu v těle adresáře
  - uživatel tedy musí mít právo zápisu **do adresáře**
  - nezáleží na tom, jestli má práva k inodu
- zároveň tím klesne počítadlo v daném inodu
- pokud počítadlo klesne na nulu, je tělo souboru odstraněno
  - takový objekt už nedává smysl, není odnikud referencován
  - dojde k uvolnění inodu i místa na disku
- to je také vysvětlení první ukázky

Ukázka: smazání souboru

Práva v adresáři

- v inode je uložena bitová mapa práv k objektu
  - SetUID, SetGID, Sticky
  - třikrát read, write, execute
- v kombinaci s vlastníkem a skupinou umožňují hlídat oprávnění
- existuje celkem 4096 kombinací (2^{12}), všechny jsou platné
  - ne všechny ale dávají v praxi smysl
- je potřeba vědět, co znamená rwx na adresáři

ROOT.CZ_ vpsFree.cz

Není adresář jako soubor

- u souboru znamená rwx: zápis, čtení nebo spouštění těla souboru
- k tělu adresáře ale přistupujeme **trochu** odlišným způsobem
  - r je čtení z těla adresáře (seznamu souborů)
  - w je zápis do těla adresáře (seznamu souborů)
  - x (search bit) umožňuje **přístup k inodům** objektů
- r dává možnost číst **názvy souborů**
  - pokud máme jen r, zjistíme jen názvy – žádné podrobnosti
- x dává právo **přistupovat k inodům** objektů v adresáři
  - pokud máme jen x, zjistíme naopak jen podrobnosti
  - musíme ale znát předem název objektu

Ukázka: práva na adresáři

Tečka a dvě tečky

- v každém adresáři se nacházejí položky `.` a `..`
- používáme je při přesunu o adresář výše: `cd ..`
- nebo pro zkrácený zápis absolutní cesty `./podadresář/`
- není to žádná „magická konstanta“, ty objekty tam skutečně jsou
- jsou to **hard linky** na současný a nadřazený adresář
  - `.` je současný adresář
  - `..` je nadřazený adresář
- bez nich bychom měli problém se ve struktuře pohybovat
  - neměli bychom zpětnou referenci pro návrat
  - systém se tedy brání jejich smazání
- jejich funkci lze ověřit porovnáním čísel inodů

Ukázka: tečka a dvě tečky

Čísla speciálních inodů

- všechny inody si nejsou rovny
- první desítka je pevně dána a má speciální význam

1. seznam vadných bloků
2. kořenový adresář
3. uživatelské kvóty
4. skupinové kvóty
5. data zavaděče
6. nesmazatelný adresář
7. resize inode
8. data pro žurnál
9. objekty vyloučené ze snapshotů
10. rezervováno pro další vlastnosti
11. první nerezervovaný, obvykle lost+found

ROOT.CZ_ vpsFree.cz

Odkazy na bloky s daty

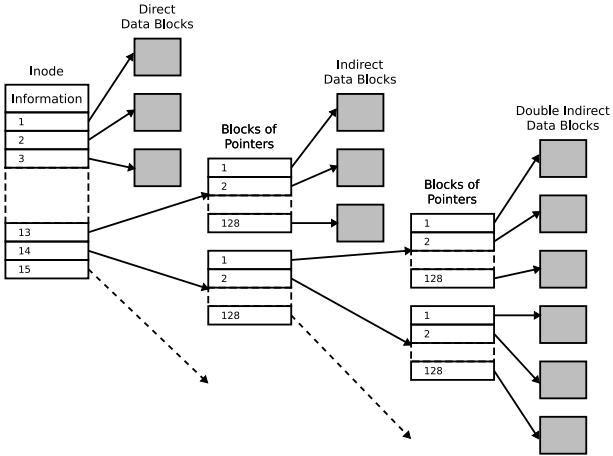
- součástí inode je seznam bloků s daty daného objektu
- moderní souborové systémy umožňují do inodu rovnou uložit data
  - `inlining` – šetří to diskové operace
  - jen pro velmi malé soubory, typicky do 60 bajtů
  - je třeba to zapnout při tvorbě souborového systému
  - hodí se to pro mnoho symbolických odkazů (fast symbolic links)
- obvykle ale inode obsahuje odkazy na bloky dat
- starší varianty ext2 a ext3 používají blokové ukazatele (block pointers)
- novější ext4 pak přešel na efektivnější systém rozsahů (extents)

Blokové ukazatele (ext2 a 3)

- každý inode obsahuje sadu ukazatelů s různou funkcí
- Direct Block Pointers: 12 přímých ukazatelů na bloky
  - nenulová hodnota je odkazem na konkrétní blok
  - pokud je soubor větší, použije se následující typ ukazatele
- Singly Indirect Block Pointer: odkazuje na blok s dalšími metadaty
  - mezi daty se rezervuje blok, do nějž se uloží seznamy dalších bloků
  - adresy jsou 32bitové a jejich počet závisí na velikosti bloku
- Doubly Indirect Block Pointer: odkaz na dvouúrovňovou strukturu
  - pokud první rozšiřující blok nestačí, použijí se dvě úrovně
  - jeden blok odkazuje na další bloky, které odkazují na data
- Triply Indirect Block Point: totéž ještě potřetí
  - pak máme tři úrovně pro opravdu velké soubory
  - zároveň tím ale roste počet operací při práci

ROOT.CZ vpsFree.cz

Schéma bloků v inodu



Rozsahy (extents)

- ext4 navyšuje velikost inodu na 256 bytů (dříve 128 bytů)
- pro uložení informací o blocích používá extenty
  - seznamy (rozsahy) navazujících bloků obsahujících data
  - jeden extent může odkazovat až na **128 MB dat**
- zlepšuje to výkon u přístupu k velkým souborům
- snižuje to problémy s fragmentací
- v každém inodu mohou být až **čtyři extenty**
- pokud je jich potřeba více, vznikne strom uložený v blocích
  - inode pak odkazuje na blok s kořenem (index node)
  - ten odkazuje na bloky s listy
  - listy pak obsahují rozsahy bloků použitých pro data

ROOT.CZ_ vpsFree.cz

Kdy se to může rozbít

- soubory se pořád mění, přibývají, zvětšují, mažou…
- každá taková operace znamená spoustu kroků
  - operace nejsou atomické, probíhají postupně
  - většina je asynchronních, nezapisuje se hned
- souborový systém se neustále snaží být konzistentní
- může ale dojít k problémům, při kterých zůstane struktura rozbitá
  - nejčastěji při selhání hardware nebo výpadku energie
- může dojít k různým typům inkonzistence
  - blok je označen jako alokovaný, ale inode na něj ještě neodkazuje
  - inode už odkazuje, ale na ještě nealokovaný blok
  - při mazání inode už neodkazuje, ale bloky jsou obsazené (únik)

Co všechno kontroluje fsck

1. superblok
   - odpovídá velikost souborového systému počtu bloků?
2. volné bloky
   - odpovídají si data v inodech a volné bloky?
3. stav inodů
   - jsou všechna data v pořádku? dávají typy objektů smysl?
4. počty odkazů na inody
   - sedí počítadla odkazů ve všech inodech?
5. duplicity
   - neodkazují některé inody na stejné bloky?
6. chybné odkazy na bloky
   - neodkazují některé inody na neplatné adresy bloků?
7. struktura adresářů
   - jsou všechny odkazované inody používané?

ROOT.CZ vpsFree.cz

Děkuji za pozornost

Otázky?

Petr Krčmář
petr.krcmar@iinfo.cz

ROOT.CZ_ vpsFree.cz