Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

KIV/ZOS
Semestrální práce

Souborový systém

14. ledna 2020 Pavel Zelenka

1 Úvod

1.1 Zadání práce

Tématem semestrální práce je práce se zjednodušeným souborovým systémem založeným na i-uzlech.

Základní funkčnost, kterou musí program splňovat. Formát výpisů je závazný.

Program bude mít jeden parametr a tím bude název souborového systému. Po spuštění bude program čekat na zadání jednotlivých příkazů s minimální funkčností viz níže (všechny soubory mohou být zadány jak absolutní, tak relativní cestou):

1. Zkopíruje soubor s1 do umístění s2.
   `cp s1 s2`
   Možný výsledek:
   `OK`
   `FILE NOT FOUND` (není zdroj)
   `PATH NOT FOUND` (neexistuje cílová cesta)

2. Přesune soubor s1 do umístění s2. `mv s1 s2`
   Možný výsledek:
   `OK`
   `FILE NOT FOUND` (není zdroj)
   `PATH NOT FOUND` (neexistuje cílová cesta)

3. Smaže soubor s1.
   `rm s1`
   Možný výsledek:
   `OK`
   `FILE NOT FOUND`

4. Vytvoří adresář a1.
   `mkdir a1`
   Možný výsledek:
   `OK`
   `PATH NOT FOUND` (neexistuje zadaná cesta)
   `EXIST` (nelze založit, již existuje)

5. Smaže prázdný adresář a1.
   `rmdir a1`
   Možný výsledek:
   `OK`
   `FILE NOT FOUND` (neexistující adresář)
   `NOT EMPTY` (adresář obsahuje podadresáře, nebo soubory)

6. Vypíše obsah adresáře a1.
   `ls a1`
   Možný výsledek:
   `-FILE`
   `+DIRECTORY`
   `PATH NOT FOUND` (neexistující adresář)

7. Vypíše obsah souboru s1.
   `cat s1`
   Možný výsledek:
   `OBSAH`
   `FILE NOT FOUND` (není zdroj)

8. Změní aktuální cestu do adresáře a1.
   `cd a1`
   Možný výsledek:
   `OK`
   `PATH NOT FOUND` (neexistující cesta)

9. Vypíše aktuální cestu.
   `pwd`
   Možný výsledek:
   `PATH`

10. Vypíše informace o souboru/adresáři s1/a1 (v jakých clusterech se nachází).
    `info a1/s1`
    Možný výsledek:
    `NAME - SIZE - i-node NUMBER - přímé a nepřímé odkazy`
    `FILE NOT FOUND` (není zdroj)

11. Nahraje soubor s1 z pevného disku do umístění s2 v pseudoNTFS.
    `incp s1 s2`
    Možný výsledek:
    `OK`
    `FILE NOT FOUND` (není zdroj)
    `PATH NOT FOUND` (neexistuje cílová cesta)

12. Nahraje soubor s1 z pseudoNTFS do umístění s2 na pevném disku.
    `outcp s1 s2`
    Možný výsledek:
    `OK`
    `FILE NOT FOUND` (není zdroj)
    `PATH NOT FOUND` (neexistuje cílová cesta)

13. Načte soubor z pevného disku, ve kterém budou jednotlivé příkazy, a začne je sekvenčně vykonávat. Formát je 1 příkaz/1řádek.
    `load s1`
    Možný výsledek:
    `OK`
    `FILE NOT FOUND` (není zdroj)

14. Příkaz provede formát souboru, který byl zadán jako parametr při spuštení programu na souborový systém dané velikosti. Pokud už soubor nějaká data obsahoval, budou přemazána. Pokud soubor neexistoval, bude vytvořen. `format 600MB`
    Možný výsledek:
    `OK`
    `CANNOT CREATE FILE`

15. Vytvoří symbolický link na soubor s1 s názvem s2.
    `slink s1 s2`
    Možný výsledek:
    `OK`
    `FILE NOT FOUND` (není zdroj)

Budeme předpokládat korektní zadání syntaxe příkazů, nikoliv však sémantiky (tj. např. cp s1 zadáno nebude, ale může být zadáno cat s1, kde s1 neexistuje).

1.2 Popis souborového systému využivající i-uzly

Soubor souborového systému je rozdělen do `N` **blokových skupin**. Každá bloková skupina obsahuje na začátku **superblok**, kde je uvedeno další rozdělení na **bitmapu**, **i-uzly** a **datové bloky**. Bitmapa obsahuje informace o zaplnění datových bloků, kdy pořadí bitu v bitmapě odpovídá pořadí datového bloku. I-uzel reprezentuje záznam souboru, kde je uvedena velikost souboru, typ souboru a odkazy na datové bloky. Datové bloky jsou pozice pevné velikosti pro ukládání dat na systém souborů.

2 Popis řešení

Aplikace je psaná v jazyce `C++`. Většina funkčností aplikace je vyčleněna v samostatném souboru, který je pojmenován podle příkazu, jenž obsluhuje.

2.1 Struktura filesystem

Struktura uchovává cestu k souboru na pevném disku, načtený superblok, kořenový adresář a současný adresář.

2.2 Struktura superblock

Struktura uchovává velikost databloku a adresy na počáteční pozice oblastí bitmapy, i-uzlů a datových bloků.

2.3 Struktura pseudo_inode

Struktura reprezentující i-uzel. Obsahuje ID, typ, počet referencí, velikost souboru, přímé a nepřímé odkazy.

2.3.1 ID i-uzlu

Ke každému souboru se váže i-uzel, který uchovává metadata souboru. Je-li uzel volný, id je rovno nule. Podle ID lze určit adresu v otevřeném souboru na pevném disku.

$$\text{s} = \text{počáteční pozice oblasti i-uzlů}$$

$$\text{n} = \text{ID i-uzlu}$$

$$\text{pozice i-uzlu} = s + (n - 1) \cdot \text{sizeof(pseudo\_inode)}$$

2.3.2 Typ i-uzlu

V práci jsou soubory tří typů, tj. adresáře, soubory, symbolické odkazy. Ve struktuře jsou reprezentovány číslem.

$$0 \ldots \text{soubor}$$

$$1 \ldots \text{adresář}$$

$$2 \ldots \text{symbolický odkaz}$$

2.3.3 Počet referencí na i-uzel

V aplikaci je vždy počet referencí jedna nebo nula. V případě implementace hardlinků by bylo možné, aby na jeden i-uzel odkazovalo několik souborů, které sdílejí obsah. Význam počtu referencí je při mazání souborů, kdy k mazání dat může dojít až se smazáním poslední reference.

2.3.4 Velikost souboru

Soubory vždy zabírají celý datový blok, pro určení konce souboru je tedy uváděna hodnota o skutečné velikosti. Při práci se souborem tedy je načteno jen tolik dat, kolik skutečně souboru náleží. Adresáře v souborovém systému vždy využijí celý datový blok.

2.3.5 Odkazy i-uzlu

I-uzel obsahuje pět přímých odkazů na datové bloky. Pro případ, že by to nedostačovalo, i-uzel obsahuje i dva nepřímé odkazy. První nepřímý odkaz je adresa datového bloku, kde jsou uloženy další přímé adresy na data souboru. Druhý nepřímý odkaz je adresa datového bloku, kde je uloženo `N` adres na dalších `N` datových bloků, až na nich jsou přímé odkazy.

2.4 Struktura directory_item

Struktura reprezentuje pojmenovaný záznam adresáře, souboru nebo symbolického odkazu s ID i-uzlu, který reprezentuje.

2.5 Formátování systému souborů

Obsluha formátování se nachází v souboru *format.cpp*. Funkce `format` má dva parametry, referenci na strukturu `filesystem`, kde je uvedena cesta k souboru a maximální možnou velikost systému souborů v bytech.

Dle velikosti systému souborů se nastaví velikost datového bloku na `1024`, `2048` nebo `4096 B`. Je brán ohled na maximální velikost souboru vůči velikosti systému souborů.

Počet i-uzlů je nastaven tak, aby byl vyšší než počet datových bloků.

Počet datových bloků je vždy číslo dělitelné 8, díky čemuž je vždy zaplněný v bitmapě celý byte.

Při formátu se vytváří i kořenový adresář, který je vždy na prvním i-uzlu a na prvním datovém bloku.

2.6 Průchod adresáři

Každý adresář obsahuje záznam `.`, kterým adresář odkazuje sám na sebe a záznam `..`, kterým odkazuje na nadřazený adresář. Kořenový adresář má oba záznamy nastaveny na sebe.

Vyhledávání dle zadané lokace provádí funkce `iNodeByLocation` v souboru *inode.cpp*. Funkce může vracet reference na inode adresáře, souboru i symbolického odkazu.

Cesta je zadávána jako textový řetězec, který je rozdělen lomítkem (tj. dle symbolu `/`). V případě absolutní cesty je lomítko zadáno na začátku a první řetězec v rozděleném poli je prázdný, nastaví se tedy kořenový adresář jako začátek průchodu. Není-li první řetězec prázdný, jako začátek průchodu je označen současný adresář.

Od procházeného adresáře se získávají všechny platné (tj. nenulové) odkazy na datové bloky. Z datových bloků jsou načteny záznamy `directory_item`, kde se hledá záznam požadovaného názvu s odkazem na další i-uzel.

V případě symbolických odkazů je na prvním přímem odkazu uložena nová cesta, rekurzivně se provede vyhledání nové cesty.

2.7 Zápis souborů

Zápis je obdobně prováděn v souborech *incp.cpp*, *cp.cpp*, *mkdir.cpp* a *slink.cpp*. Nejnáročnější je implementace pro příkaz `incp`, kde dochází ke kopírování dat z pevného disku. Před zápisem dojde ke zjištění velikosti načítaného souboru, přepočet dostupných databloků z bitmapy, omezení maximálního počtu databloků na soubor a zajištění volného i-uzlu.

Soubor se načítá z pevného disku do pole datového typu `char` o velikosti jednoho datového bloku, vždy po načtení do pole se provede zápis do systému souborů a pole se uvolní pro načítání další části.

Po načtení souboru se vytvoří v nadřezeném adresáři nový záznam `directory_item`, který odkazuje na i-uzel souboru.

2.8 Odstranění souborů

Odstranění souboru i adresářů je implementováno souborem `rm.cpp`. Odstranit lze pouze prázdné adresáře. Kořenový adresář smazat nelze, protože neexistuje nadřazený adresář, respektive odkazuje sám na sebe. Při odstraňování nelze procházet skrze symbolické odkazy. Datové bloky se mažou až při opětovném zápisu, při mazání dochází pouze k vynulování odpovídajícího bitu v bitmapě a nastavení použitého i-uzlu na nulové hodnoty.

2.9 Čtení souborů

Čtení souborů se provádí například v souborech *outcp.cpp*, *cat.cpp* či *info.cpp*. Funkce `iNodeByLocation` obstará požadovaný i-uzel, může procházet i skrze symbolické odkazy. V druhém kroku se získají skrze metodu `usedDatablockByINode` ze souboru *datablock.cpp* všechny použité adresy, které budou načteny do vektoru. Výpis se provádí procházením adres jedné po druhé a vkládáním do bufferu o velikosti jednoho datového bloku, který se následně vypisuje.

2.10 Omezení zvoleného řešení

Byty souborového systému jsou reprezentovány datovým typem `int32_t`, který může nabývat maximální hodnoty `2 147 483 647`. To omezuje maximální velikost souborového systému na `2 GB`. Tuto hodnotu by bylo možné zdvojnásobit použitím neznamenkového datového typu `uint32_t`. Pro ještě větší velikosti je možné použít datobé typy z knihovny `multi-precision library`.

Velikost databloku je závislá a pevně daná dle velikosti souborového systému. Souborový systém do velikosti `100 MB` má databloky o velikosti `1 kB`, dále do velikosti `600 MB` jsou databloky o velikosti `2 kB`, následně pro větší souborové systemy je zvolena velikost `4 kB`.

Maximální možná velikost souboru se odvíjí od velikosti databloku. U nepřímých odkazů se do databloků ukládají adresy o velikosti `4 kB`, protože je pro reprezentaci použit datový typ `int32_t`. Dle vzorce lze spočítat maximální možnou velikost souboru.

$$\mathrm{d} = \text{velikost databloku}$$

$$5 \cdot d + \left(\frac{d}{4}\right) \cdot d + \left(\frac{d}{4}\right)^2 \cdot d$$

Maximální možná velikost souboru			
datablok	**B**	**kB**	**MB**
1024	67 376 128	65 797	64,25
2048	537 929 728	525 322	513,01
4096	2 149 601 280	2 099 220	2 050,01

Tabulka 1: Maximální možná velikost souboru dle velikosti databloku.

3 Uživatelská dokumentace

3.1 Požadavky

Aplikace je primárně určena pro operační systém **GNU/Linux**. Pro rozbalení archívu a překlad aplikace byly použity nástroje:

- tar,
- cmake,
- make,
- gcc.

3.2 Postup pro GNU/Debian

1. Nainstaluje do systému potřebné balíčky tar, cmake, make a gcc.
   `sudo apt install tar cmake make gcc`
2. Rozbalení archívu s aplikací.
   `tar xvzf PseudoFS.tar.gz`
3. Přesunutí do adresáře aplikace.
   `cd PseudoFS/src/build`
4. Provede překlad aplikace.
   `cmake .. && make`
5. Spuštění aplikace.
   `./PseudoFS`

3.3 Ovládání

Aplikace se ovládá příkazy tak, jak jsou uvedeny v zadání práce. Pro ukončení aplikace slouží navíc příkaz `q`.