

Souborový systém

KIV/ZOS - Semestrální práce

student: Jan Hinterholzinger
studijní číslo: A19B0052P
email: hintik@students.zcu.cz
datum: 10. 01. 2022

1 Popis zadání SP

Program simulující souborový systém založený na i-uzlech. Program při spuštění očekává jako parametr název souboru, s kterým bude program pracovat. Po spuštění bude program čekat na zadání jednotlivých příkazů s minimální očekávanou funkčností (popisána v originálním dokumentu zadání). Všechny cesty k souborům mohou být zadány jak absolutní, tak relativní cestou.

2 Analýza problému

2.1 I-uzly

I-uzly (inode) je datová struktura, která uchovává informace (metadata) o souborech a adresářích, která je využívána ve spoustě souborových systémů. Informace, které

Soubor informací, které i-uzel o souboru obsahuje může být různý. Typicky však uchovává velikost, typ, práva k souboru, časové značky, atd. Nejpodstatnější položkou však jsou odkazy na datové bloky.

Odkazy na datové bloky jsou rozdělené na přímé, kdy v daném data bloku jsou přímo data souboru, a nepřímé. Nepřímé odkazy vedou na data bloky, které mají v sobě další seznam data bloků, ve kterých už mohou být data souboru. Tyto odkazy mohou mít více úrovní a tím navyšovat maximální velikost souboru.

I-uzly však neobsahují název souboru. Název je uložen v jednotlivých datových blocích adresáře, kdy je spojen ve dvojici s identifikátorem i-uzlu. Také díky tomu je možné odkazovat na jeden soubor ze dvou míst a tvořit tzv. pevné odkazy.

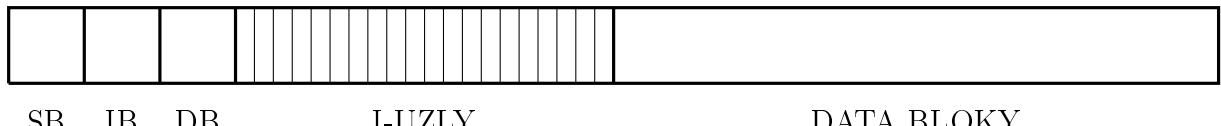
Díky tomuto systému je proto jednoduché pro zadání cestu v traverzovat přes složky k souborům, tak že hledáme v data bloku název souboru a tím se dostaneme na i-uzel další úrovně. Toto opakujeme, dokud nezískáme i-uzel hledaného souboru.

2.2 Souborový systém

Samotný souborový systém potřebuje mít struktury, aby bylo možné v něm jednoduše vyhledávat a procházet. Jak již bylo zmíněno, pro ukládání dat a informací o souboru jsou využívány struktury i-uzlu a data bloky. Avšak pro správu a práci potřebujeme více částí.

Potřebujeme navíc nějakou evidenci, který i-uzel resp. data blok je volný a který je obsazený. Jedná se o binární hodnotu pro každý jednotlivý i-uzel nebo data blok, proto je vhodné si uchovávat v souborovém systému bitmapy i-uzlů (IB) a data bloků (DB).

Také potřebujeme se v souborovém systému rychle pohybovat, a případně uložit nějaké informace o celém systému, proto si zavedeme blok paměti a budeme ho nazývat super blok (SB). Pro rychlý pohyb budeme v tomto bloku uchovávat adresy začátků všech ostatních částí souborového systému. Zároveň si zde můžeme uložit různé informace, které spadají pro celý souborový systém. Jelikož super blok obsahuje adresy ostatních částí, budeme jej pravděpodobně načítat jako první, proto se hodí ho umístit na začátek struktury.



Obrázek 1: Schéma souborového systému

3 Implementace

Program je implementovaný v programovacím jazyku C. Při startu programu se program pokusí otevřít soubor z parametrem zadané cesty. Pokud neuspěje, bude program ve stavu, kdy očekává pouze příkazy `exit` pro ukončení programu, `load` pro načtení příkazů ze souboru a `format` pro zformátování souboru.

Program si načítá ze souboru pouze ty data, která aktuálně potřebuje. Po ukončení operace s nimi je uloží zpět na úložiště. Program tedy je v konzistentním stavu vždy, když je dokončen příkaz a než se spustí další příkaz.

Program si uchovává strukturu `file_system`, kterou si vede po celou dobu programu a používá se téměř ve všech funkcích. Obsahuje informace, které jsou potřeba pro splnění požadavků, například odkaz na soubor, spojový seznam pro ukládání cesty nebo identifikátor aktuálního adresáře. Také obsahuje strukturu super bloku, protože se v průběhu programu super blok nemění a je tedy zbytečné ho pokaždé načítat znova. Super blok se při ukončování programu nezapisuje zpět na úložiště.

Evidence zabraných i-uzlů a data bloků je vedena pomocí bitmapy, která je používána pomocí bitových posunů a bitových operátorů.

4 Seznam příkazů

4.1 Zkopírování souboru

Systém projde všechny (nenulové) data bloky souboru a vytvoří identickou kopii příkazem `cp <zdrojový soubor> <cílový soubor>`.

Příklad:

```
$ cp /home/petr/hesla.txt /home/me/pass.txt
```

4.2 Přemístění souboru

Příkaz pro přemístění (nebo přejmenování) souboru. Systém odstraní dvojici jména a čísla i-uzlu z rodičovského adresáře a do cílového adresáře vloží novou dvojici s novým jménem, ale stejným identifikátorem i-uzlu. Použití: `mv <zdrojová cesta> <cílová cesta>`

Příklad:

```
$ mv /home/me/ransom.sh /home/petr/fotka.jpg
```

4.3 Odstranění souboru

Pro vymazání souboru ze souborového systému použijeme příkaz `rm <cesta k souboru>`.

Příklad:

```
$ rm .. / petr / nakup . txt
```

4.4 Vytvoření adresáře

Vytvoření adresáře provedem příkazem `mkdir <cesta k nové složce>`

Příklad:

```
$ mkdir .. / jenda /
```

4.5 Odstranění adresáře

V systému je možné odstraňovat pouze prázdné adresáře příkazem `rmdir <cesta k adresáři>`. Nelze odstranit kořenový adresář a adresář, ve kterém se ukazatel aktuálního adresáře nachází.

Příklad:

```
$ rmdir /home/ petr /
```

4.6 Vypsání obsahu adresáře

K zjištění obsahu adresáře poslouží příkaz `ls {cesta k adresáři}` (parametr je nepovinný), který vypíše seznam prvků složky včetně jejich typů („+“ - adresář, „-“ - soubor).

Příklad:

```
$ ls /home/
```

Příklad:

```
$ ls
```

4.7 Vypsání obsahu souboru

Vypsání obsahu souboru je možné pomocí příkazu `cat <cesta k souboru>`. Příkaz projde veškeré datové bloky a v nezměněné (kromě správného oříznuté posledního data bloku) podobě je vypíše na výstup.

Příklad:

```
$ cat /home/pepa/ soubor . txt
```

4.8 Změna adresáře

V programu lze procházet mezi adresáři pomocí příkazu `cd <cesta k adresáři>`, kdy se následně změní počátek všech zadaných relativních cest.

Příklad:

```
$ cd /home/pepa /
```

4.9 Vypsání aktuální cesty

Program si zaznamenává změny adresáře pro vypisování aktuální absolutní cesty. Tato cesta se jednak vypisuje před znakem dolaru (\$) při ručním zadáváním příkazů nebo při použití příkazu `pwd`. Cesta je vnitřně uložena ve struktuře spojového seznamu.

Příklad:

```
$ pwd
```

4.10 Vypsání informací o souboru/adresáře

Příkaz vypíše informace z i-uzlu ze zadaného souboru (resp. adresáře). Vypíše se název, velikost, kterou soubor zabírá v souborovém systému, číslo i-uzlu a seznam data bloků, kde jsou uloženy data. Formát příkazu: `info <cesta k souboru>`.

Příklad:

```
$ info /home/soubor.txt
```

4.11 Nahrání souboru do souborového systému

Pro práci v souborovém systému potřebujeme samozřejmě nějaká data, tedy soubory. Soubory se do systému přidávají příkazem `incp <externí zdroj> <cílová cesta v systému>`.

Příklad:

```
$ incp soubor.txt /home/soubor.txt
```

4.12 Nahrání souboru ze souborového systému

Ze souborového systému lze soubory i nahrať zpět do souborového systému Vašeho zařízení pomocí příkazu `outcp <zdrojová cesta v systému> <externí výstup>`. Výstupní soubor je identický se souborem, který byl původně vložen.

Příklad:

```
$ outcp /home/soubor.txt soubor.txt
```

4.13 Načtení seznamu příkazů

Souborový systém je schopný příkazy vykonávat dávkově tím, že je vložíme do souboru ve formátu 1 příkaz odpovídající 1 řádce a soubor načteme pomocí příkazu `load <cesta k souboru>`. Program vykoná postupně každý příkaz a po ukončení dávky je umožněno uživateli zadávat příkazy ručně.

Příklad:

```
$ load adresar/soubor.txt
```

4.14 Zformátování souborového systému

Souborový systém je možné (a pro používání potřeba) zformátovat. Pro zformátování je vyčleněn příkaz `format <velikost souboru>`, kdy `velikost souboru` se zadává ve formátu čísla a jednotky (viz. tabulka 1).

Při formátování se vytvoří soubor (pokud do té doby neexistoval) a vyplní se na zadanou velikost. Po úspěšném formátování je možné zadávat příkazy pro obsluhu souborového systému.

Seznam jednotek	
B	1 B
kB	1 000 B
kiB	1 024 B
MB	1 000 000 B
MiB	1 048 576 B
GB	1 000 000 000 B
GiB	1 073 741 824 B

Tabulka 1: Seznam možných jednotek

Příklad:

```
$ format 50MB
```

4.15 Vytvoření pevného odkazu

Pevný odkaz je zjednodušeně prvek složky, který odkazuje na i-uzel, na který již někde vyskytuje odkaz. Pro vytvoření pevného odkazu je možné použít příkaz `ln <existující soubor> <cesta k vytvoření odkazu>`. Pevný odkaz nelze vytvářen pro adresáře.

Příklad:

```
$ ln /home/soubor.txt zkratka.txt
```

4.16 Opuštění programu

Pro opuštění programu je možné kdykoli zadat příkaz `exit`, který uzavře souborový systém a ukončí program.

Příklad:

```
$ exit
```

5 Spuštění programu

Program je dodán v archívu ZIP a je umístěn v adresáři `src`. Program není dodán se zkompilovaným spouštěčem ani připraveným Makefilem. Pro vytvoření souboru Makefile je potřeba použít program `cmake`. V adresáři `src` zahájíme zadávání těchto příkazů.

Vytvoření souboru Makefile:

```
$ cmake -B ./build /
```

Přepneme se do adresáře build:

```
$ cd ./build
```

Zkompilujeme program a vytvoříme spouštěč:

```
$ make
```

Nyní je možné spustit program s parametrem souboru, kam se bude souborový systém ukládat.

Spuštění programu:

```
$ ./sp hintik.fs
```

6 Závěr

Program pracuje se souborovým systémem využívající i-uzly. Jsou implementované všechny zadané příkazy a navíc se snaží kontrolovat jejich syntax. Přijímá absolutní a relativní cesty a dokáže operovat se soubory tak, že při vyjmutí souborů ze souborového systému jsou v identickém stavu jako vstupní soubory. Z důvodu neideálnímu počátečnímu rozvržení a snaze o zjednodušení nejsou některé části programu dostatečně optimalizovány.