

Souborový systém založený na i-uzlech

Katedra informatiky a výpočetní techniky
Semestrální práce z předmětu KIV/ZOS

Štěpán Faragula
A21B0119P
farag844@students.zcu.cz

4. ledna 2024

Obsah

1	Zadání	2
2	Struktura souborového systému	4
3	Programátorská dokumentace	5
4	Uživatelská příručka	6
5	Závěr	7

Kapitola 1

Zadání

Tématem semestrální práce bude práce se zjednodušeným souborovým systémem založeným na i-uzlech. Vaším cílem bude splnit několik vybraných úloh.

Program bude mít jeden parametr a tím bude název Vašeho 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`
2. Přesune soubor `s1` do umístění `s2`, nebo přejmenuje `s1` na `s2`
`mv s1 s2`
3. Smaže soubor `s1`
`rm s1`
4. Vytvoří adresář `a1`
`mkdir a1`
5. Smaže prázdný adresář `a1`
`rmdir a1`
6. Vypíše obsah adresáře `a1`
`ls a1`
`ls`

7. Vypíše obsah souboru s1
`cat s1`
8. Změní aktuální cestu do adresáře a1
`cd a1`
9. Vypíše aktuální cestu
`pwd`
10. Vypíše informace o souboru/adresáři s1/a1 (v jakých clusterech se nachází)
`info a1`
`info s1`
11. Nahraje soubor s1 z pevného disku do umístění s2 ve vašem FS
`incp s1 s2`
12. Nahraje soubor s1 z vašeho FS do umístění s2 na pevném disku
`outcp s1 s2`
13. Načte soubor z pevného disku, ve kterém budou jednotlivé příkazy, a začne je sekvencně vykonávat. Formát je 1 příkaz na 1 řádek
`load s1`
14. Formát souboru, který byl zadán jako parametr při spuštění programu na souborový systém dané velikosti. Pokud už soubor nějaká data obsahoval, budou přemazána. Pokud soubor neexistoval, bude vytvořen.
`format 600MB`
15. Vytvoří hardlink na soubor s1 s názvem s2. Dále se s ním pracuje očekávaným způsobem, tedy např. `cat s2` vypíše stejný obsah jako `cat s1`.
`ln s1 s2`

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

Každý název bude zabírat právě 12 bytů (do délky 12 bytů doplníme `'\0'`).

Dále předpokládáme, že adresář se vždy vejde do jednoho clusteru (limituje nám počet položek v adresáři).

Kapitola 2

Struktura souborového systému

Souborový systém založený na i-uzlech reprezentuje každý soubor či adresář jedním i-uzlem (anglicky index node). Jedná se o strukturu obsahující metadata a reference na datové bloky souboru. Tyto odkazy jsou rozděleny na přímé a nepřímé, kde přímé ukazují na datový blok s obsahem souboru a nepřímé na čísla bloků ve kterých jsou uloženy data souboru. Nepřímé odkazy mohou být víceúrovňové, kde každá úroveň víc bude ukazovat na další nepřímé odkazy. V této práci obsahuje každý i-uzel 5 přímých odkazů, 1 nepřímý 1. úrovně a 1 nepřímý 2. úrovně.

Adresáře ukládají na datový blok dvojici názvu souboru a index i-uzlu pro každý soubor co obsahují. Dále obsahují dvojici reprezentující sebe sama '.' a rodičovský adresář '..'.

Celý souborový systém pracuje s přiděleným místem na disku. V případě semestrální práce je tento prostor simulován textovým souborem zadané velikosti. Aby mohl být systém snadno uložen a načten, zapisuje data do souboru v určitém pořadí, které je znázorněno na obrázku 2.1. Význam jednotlivých bloků je následující:

- Superblock = metadata souborového systému
- Bitmapa i-uzlů = zobrazuje volné/zabrané indexy i-uzlů
- Bitmapa datových bloků = zobrazuje volné/zabrané indexy datových bloků
- Oblast i-uzlů = zde se ukládají datové struktury i-uzlů
- Oblast datových bloků = zde se ukládají konkrétní data souborů/adresářů

Superblock	Bitmapa i-uzlů	Bitmapa datových bloků	Oblast i-uzlů	Oblast datových bloků
------------	----------------	------------------------	---------------	-----------------------

Obrázek 2.1: Struktura souborového systému

Kapitola 3

Programátorská dokumentace

Počet i-uzlů je pevně nastaven na 1024, je tak nejlépe využít paměťový prostor příslušné bitmapy. Velikost datového bloku je nastavena na 512 bytů. Vzhledem k počtu referencí co může i-uzel obsahovat je maximální velikost souboru zhruba 8,5 MB. Jelikož názvy souborů jsou omezeny na 12 znaků a pro reference je použit 32bitový integer, datový blok adresáře může obsahovat maximálně 32 odkazů na datové bloky.

Práce je vytvořena v jazyce C/C++. Požadavky uživatele jsou vyhodnocovány v nekonečné smyčce pomocí funkce `parse_input()`. Struktura programu je rozdělena mezi několik souborů obsahující třídy a funkce zabývající se určitou problematikou. Ve zkratce se jednotlivé soubory zabývají následujícím:

- **Constants** - konstanty programu
- **Main** - spouští souborový systém, přijímá vstup uživatele
- **InputParser** - zpracovává vstup, předává data k vykonání požadavku
- **FileSystem** - obsahuje jednotlivé příkazy které uživatel volá, vykonává operace nad virtuálním souborovým systémem
- **Superblock** - metadata souborového systému
- **Bitmap** - bitmapa volných indexů
- **IndexNode** - i-uzel
- **Directory** - datový blok adresáře
- **DirectoryItem** - položka adresáře
- **ReferenceBlock** - datový blok obsahující odkazy na jiné datové bloky

Kapitola 4

Uživatelská příručka

Pro překlad a spuštění aplikace je nutné mít nainstalované programy `g++`, `cmake` a `make`. Program očekává jeden parametr při spuštění, a to cestu k souboru obsahující data souborového systému. Příklad spuštění je následující:

```
./ZOS_Semestralka myFileSystem
```

Pokud soubor existuje, zkusí data načíst. Pokud ne, vyžaduje od uživatele o zavolání příkazu `format`, který následně soubor vytvoří. Aplikace se řádně ukončí příkazem `exit` či zasláním signálu `SIGINT`.

Tabulka 4.1 obsahuje všechny příkazy, které lze nad souborovým systémem vykonat.

<code>cp s1 s2</code>	Zkopíruje soubor <code>s1</code> do umístění <code>s2</code>
<code>mv s1 s2</code>	Přesune soubor <code>s1</code> do umístění <code>s2</code> , nebo přejmenuje <code>s1</code> na <code>s2</code>
<code>rm s1</code>	Smaže soubor <code>s1</code>
<code>mkdir a1</code>	Vytvoří adresář <code>a1</code>
<code>rmdir a1</code>	Smaže prázdný adresář <code>a1</code>
<code>ls a1</code>	Vypíše obsah adresáře <code>a1</code>
<code>cat s1</code>	Vypíše obsah souboru <code>s1</code>
<code>cd a1</code>	Změní aktuální cestu do adresáře <code>a1</code>
<code>pwd</code>	Vypíše aktuální cestu
<code>info a1</code>	Vypíše informace o souboru <code>s1</code>
<code>incp s1 s2</code>	Nahraje soubor <code>s1</code> z pevného disku do umístění <code>s2</code> ve vašem FS
<code>outcp s1 s2</code>	Nahraje soubor <code>s1</code> z vašeho FS do umístění <code>s2</code> na pevném disku
<code>load s1</code>	Načte soubor s příkazy a začne je sekvenčně vykonávat
<code>format 600MB</code>	Připraví strukturu souborového systému o dané velikosti
<code>ln s1 s2</code>	Vytvoří hardlink na soubor <code>s1</code> s názvem <code>s2</code>
<code>exit</code>	Řádně ukončí aplikaci

Tabulka 4.1: Dostupné příkazy souborového systému

Kapitola 5

Závěr

Souborový systém založený na i-uzlech splňuje všechny požadavky zadání. Umožňuje základní manipulaci se soubory a adresáři, kterou lze od souborového systému očekávat. Neumožňuje však vytvářet a upravovat obsah souborů. Tato funkcionality by vyžadovala rozhraní textového editoru, což by bylo nad rámec řešení semestrální práce.

Možné vylepšení by mohlo být v hledání volného místa v bitmapě, jelikož nyní se vyhledává jedno po druhém nezávisle na sobě a u načítání velkých souborů tento přístup způsobuje zpomalení. Namísto nezávislého naivního hledání by se bitmapa mohla prohledat pouze jednou a vrátit seznam s potřebným počtem volných míst.