Semestrání práce

z předmětu ZOS

Virtuální souborový systém

Jan Rádl

19. prosince 2021

Obsah

Kapitola 1

Zadání

1.1 Téma

Tématem semestrální práce bude práce se zjednodušeným souborovým systémem založeným na i-uzlech. Vaším cílem bude splnit několik vybraných úloh. Základní funkčnost, kterou musí program splňovat. Formát výpisů je závazný. 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)

- Maximální délka názvu souboru bude 8+3=11 znaků (jméno.přípona) + \0 (ukončovací znak v (C/C++), tedy 12 bytů.
- Každý název bude zabírat právě 12 bytů (do délky 12 bytů doplníte \0 - při kratších názvech)

1.2 Příkazy

1. cp - kopíruje soubory
2. mv - přesouvá soubory
3. rm - maže soubory
4. ln - vytvoří hard link na soubor
5. cat - vypíše obsah souboru jako sekvenci charů
6. mkdir - vytváří adresář
7. rmdir - ruší prázný adresář
8. ls - vypíše obsah adresáře

9. cd - změní aktuální adresář
10. pwd - vypíše cestu od root adresáře k aktuálnímu adresáři
11. info - vypíše informace do daném i-uzlu
12. incp - nahraje soubor do vfs
13. outcp - vytvoří kopii souboru z vfs do domovkého souborového systému
14. load - začne vykonávat příkazy ze zadaného souboru
15. format - provede zformátování vfs na požadovanou velikost v MB

Kapitola 2

Analýza úlohy

2.1 I-uzlový souborový systém

2.1.1 I-uzel(i-node)

Základní jednotka souborového systému obsahující všechny podstatné informace o datech souboru nikoliv však jeho jméno.Jmenovitě:

- unikátní identifikátor i-uzlu
- typ souboru (sobor/adresář)
- velikost soboru
- počet odkazů ukazující na tento soubor
- kolekce ukazatelů na data

Přímé adresování

I-uzel obsahuje přímo adresu data bloku.

Inline adresování

Nekteré moderní soborové systémy dovolují malé množstvý dat ulozit přímo v inode struktůře místo data bloku. Tento styl je pro tuto implementaci nevhodný kvůli poměrně malé velikosti i-uzlu.

Nepřímé adresování n řádu

Obdobně jako u nepřímého adresovaní pro proměné tak i zde je uložena pouze adresa na datový block obsahující odkazy o jeden řád nižší a pokud řád dosáhne 0 tak daný odkaz opět jako u přímého adresování obsahuje data souboru.

2.1.2 Adresářový zázman (Dir item)

Je další velice důležitou součástí toho systému ukládání souboru, protože dovoluje soubory ve souborovém systému pojmemovávat. Jedná se o jednoduchou strukturu obsahují jméno souboru a unikátní identifikátor i-uzlu stímto názvem.

2.1.3 SuperBlock

Je první struktura v souboru, která obsahuje informace pro zavedení a obsluhu daného soborového systému. Určuje rozdělení paměti na 4 části. Obsahuje:

- celkovou velikost disku
- velikost data bloku
- počet inodu
- počet data bloků
- adresu pole bitů reprezentují použitých inodu
- adresu pole bitů reprezentují použitých data bloků
- adresa prvního i-uzlu (adresa i-uzlové části systému)
- adresa prvního data bloku (adresa datatové části)

Kapitola 3

Popis implementace

3.1 Prostředí

Pro implementaci jsem si vybral programovací jazyk C++(CPP) v jeho podobě definované v jeho standartu 17(c++17). Pro jednoduché sestavení využívám nástroj CMake. Program je členěn do 2 částí:

- inodefs - knihovna, která simuluje samotný souborový systém
- fsterminal - slouží pro obsluhu systému pomocí příkazů

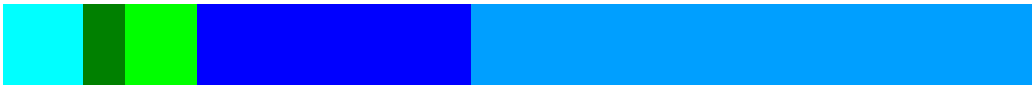
3.2 Řešení

3.2.1 Struktura souboru

Samotný soubor obsahují filesystem je rozdělen do 5 částí:

1. SuperBlock
2. I-uzlové bitové pole
3. Data blokové bitové pole
4. Pole I-uzlů
5. Pole data bloků

Takto může vypadat rozložení pro předpokladané využití souboru:



Obrázek 3.1: Ilustrace rozložení souborového systému

3.2.2 Výpočet rozložení

Velikost superbloku(SB_s) je stálá, ale ostatní bloky jsou vázené na jiné bloky nebo na využitelnou velikost souboru. Ze zadané celkové velikosti souboru(D_s) se odečte velikost superbloku a 2 byty propřípad, že počety bloků a inodů nebude dělitelný 8. Dále potřebuje poměr i-uzlů k data blokům v procentech(P_{ib}) a velikost bloku(B_s) nacházející se v souboru `config.hpp` a velikost i-uzlu(I_s). Poté se spočítá počet bloků(B_{cnt}) pro tuto velikost podle:

$$B_{cnt} = \frac{D_s - SB_s - 2}{\frac{P_{ib}}{8} + \frac{1}{8} + P_{ib} \cdot I_s + B_s}$$

Jednotlivé části čitatele odpovídají velikostem jednotlivých bloků po přenásobení B_{cnt} v bytech.

3.2.3 Bitové pole

Občas bývá také pojmenováno bitmapa. Moje implementace využívá implementaci bitSetu v cpp pro obsluhu bitových úprav a vlastního iterátoru. Logická přestava o lineárním poli bitů se neschoduje s implementací. Reálně je toto pole je tvořeno jako posloupnost 0..n bytů, kde jsou bity v bytu jsou LSB.

	0								1							
R:	07	06	05	04	03	02	01	00	15	14	13	12	11	10	09	08
L:	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15

Obrázek 3.2: Logická vs Reálná

3.2.4 Alokace a přidávání

V celém programu je hodnota 0 je považovaná jako prázná/neplatná hodnota, až na root prvek, který této hodnoty nabývá jak i-uzel tak datablock a je to jediné legální místo, které tu to hodnotu může nabývat.

AddPointer

Funkce která pro zadaný i-uzel přidá ukazatel na zadaný data block. Pokud program běží v debug konfiguraci, tak při přetečení adresovatelného postoru i-uzlem ukončí program assetem a pokud je zapnuté logování a tak errorem.Tato situace je logická chyba, která nelze nijak vyřešit, protože její vyřešení by vyžadovalo rozšíření adresovatelného prostoru. V release režimu je v i-uzlu uloženo maximum adresovatelných dat a zbytek je zapsán, ale je nedostupných.

AlocateX

Je dvojice funkcí, kde X je nahrazeno (inode/datablock), která projte příslušné bitové pole a pokud narazí na volný prvek, tak ho zabere a vrátí jeho indentifikátor.

3.2.5 Dealokace a odstranění

freeX

Obdobně jako AlocateX, ale jedná se o inverzní operaci, která navíc dané místo naplní opakujícím se znakem \0.

Remove dir item

Je funkce pro odstranění záznamu adresáře z rodičovkého adresáře pokud daný záznam je nalezen, tak je odstraněn a nahrazen posledním záznamem v tomto bloku a pokud je to zároveň poslední záznam v tomto bloku tak tento block zůstane alokovaný, ale bude prázný. Regenerace na úrovni bloků by byla příliš nákladná a navíc by zanesla do systému problém s nespojitostí datových ukazatelů v i-uzlu.

Kapitola 4

Uživatelská příručka

4.1 Překlad

Pro sestavení je potřeba využít nástoje CMake a díky tomu máte tyto možnosti:

- Logovani - vytvoření souboru main.log obsahující informace o běhu programu podle nastaveni `LOGLEVEL` v `log.hpp`
- Debug build - sestaveni programu s kontrolou stavu programu za běhu
- Release build - výchozí verze programu, která nechrání proti erroru, ale přesto mohou nastat

Příklad sestavení pro normální běh programu:
`cmake -S <adresář projektu> -B <adresář sestavení>`
Sestavení v debug modu:
`cmake -DCMAKE_BUILD_TYPE=Debug -S <adresář projektu> -B <adresář sestavení>`
Sestavení s logováním:
`cmake -DLOGFLAG=ON -S <adresář projektu> -B <adresář sestavení>`
Poté můžeme zavolat náš systémový nástroj pro setavení projektu nad adresářem setavením. Následně v `adresářem setavení/app` můžeme najít sestavený program pro obsluhu souborového pomocí terminálu.