FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce KIV/ZOS

Souborový systém založený na pseudoFAT

Student: Adam Míka
Osobní číslo: A22B0319P
Email: mikaa@students.zcu.cz
Datum: 5. prosince 2024

Obsah

Seznam obrázků

Listings

Reference

[1] Zdroj Obrázku 1: https://philipstel.wordpress.com/2010/08/04/dictionary-based-algorithm-lempel-ziv-welch/

1 Zadání

Hlavní cíle: Zbytek zadání zde: *PDF*

2 Analýza úlohy

Cílem semestrální práce je implementace zjednodušeného souborového systému na bázi pseudoFAT, který umožní uživatelům provádět základní operace se soubory a adresáři. Zadané úlohy lze rozdělit do několika hlavních kategorií podle funkcionalit, které systém poskytuje. V této sekci analyzujeme požadavky na jednotlivé příkazy a jejich význam pro souborový systém.

2.1 Základní operace

- **Kopírování souboru (cp)**: Tento příkaz umožňuje kopírovat soubor z jednoho umístění (zdroj) do druhého (cíl). Součástí implementace je nutnost validace cesty k cíli i zdroji. Příkaz musí detekovat chyby, jako například neexistující zdroj nebo cíl.
- **Přesunutí nebo přejmenování souboru (mv)**: Tento příkaz umožňuje přesunout soubor z jedné cesty do jiné, nebo přejmenovat soubor. Klíčovou částí je validace, zda cílová cesta existuje a zda nedochází ke konfliktu názvů.
- **Odstranění souboru (rm)**: Umožňuje mazat konkrétní soubory. Důležité je validovat, zda je mazán skutečně soubor (ne adresář), a ověřit existenci souboru.
- **Vytvoření adresáře (mkdir)**: Slouží k vytvoření nového adresáře. Příkaz musí ověřit, zda již neexistuje adresář nebo soubor se stejným názvem v cílové cestě.
- **Odstranění prázdného adresáře (rmdir)**: Tento příkaz odstraňuje pouze prázdné adresáře. Je nutné kontrolovat, zda je adresář skutečně prázdný, a odlišit situaci, kdy je na vstupu zadán soubor místo adresáře.

2.2 Rozšířené operace

- **Načítání příkazů ze souboru (load)**: Tento příkaz umožňuje provést příkazy uložené v textovém souboru. Formát vstupního souboru musí být validní a každý příkaz by měl být zpracován sekvenčně.

- **Formátování souborového systému (format)**: Příkaz vytvoří nový souborový systém na zadané velikosti. Při implementaci je třeba zohlednit inicializaci tabulek FAT a alokaci prostoru pro datové bloky.
- **Kontrola integrity souborového systému (check)**: Zajišťuje ověření konzistence struktury souborového systému, jako jsou chybné nebo ztracené bloky, nekonzistentní tabulky FAT a správnost datových struktur.
- **Poškození souborového systému (bug)**: Simuluje poškození systému, aby bylo možné testovat příkaz *check*.

2.3 Hodnocení a očekávané výsledky

Při implementaci a testování systému je nutné dodržet následující požadavky:

- Funkční a korektní zpracování všech příkazů.
- Validace vstupů a robustní detekce chybových stavů.
- Testování chování systému při chybách (např. použití *bug* a následné ověření příkazem *check*).
- Dokumentace výsledků a výpisů příkazů v předepsaném formátu.

2.4 Funkce FAT tabulky

FAT (File Allocation Table) je jádrem správy souborového systému a hraje zásadní roli při sledování alokace dat na disku. Princip FAT tabulky spočívá v udržování seznamu propojených bloků dat (clusterů), které tvoří jednotlivé soubory či složky. Každý cluster má odpovídající záznam v tabulce, kde je uvedeno, zda:

- je cluster součástí souboru/složky a na který následující cluster odkazuje,
- je cluster označen jako konec souboru (`FAT_FILE_END`),
- je cluster volný pro použití (`FAT_UNUSED`),
- je cluster poškozený (`FAT_BAD_CLUSTER`).

Tento systém umožňuje efektivní navigaci mezi jednotlivými částmi dat souboru a optimalizuje správu volného prostoru na disku.

2.5 Vlastnosti FAT tabulky v systému

- **Jednoduchost:** FAT tabulka je implementována jako jednorozměrné pole čísel, kde index pole odpovídá číslu clusteru. Hodnota uložená na daném indexu určuje další cluster v řetězci nebo speciální stav (např. `FAT_FILE_END`).
- **Flexibilita:** Soubory mohou být uložené v nepropojených clusterech. Pokud je soubor fragmentovaný, tabulka umožňuje jeho rekonstrukci díky propojení mezi clustery.
- **Omezení:** Fragmentace může vést ke sníženému výkonu při práci s velkými soubory, protože systém musí neustále přeskakovat mezi clustery.

2.6 Správa poškození a kontrola integrity

Pro kontrolu integrity souborového systému je využíván příkaz `check`, který identifikuje poškozené clustery (`FAT_BAD_CLUSTER`) a zjišťuje, zda nejsou součástí žádného řetězce. Příkaz `bug` umožňuje simulaci chyb tím, že označí cluster jako poškozený. Takto je možné testovat robustnost systému a funkčnost kontrolních mechanismů.

2.7 Ilustrace funkčnosti

Například soubor rozdělený do dvou clusterů:

- `FAT[0] = 1` (první cluster ukazuje na druhý),
- `FAT[1] = FAT_FILE_END` (konec souboru).

Tento přístup zajišťuje jednoduchou správu dat i v případě, že disk obsahuje fragmentované soubory. Pokud by se cluster označený jako `FAT_BAD_CLUSTER` stal součástí řetězce, příkaz `check` tuto chybu identifikuje a upozorní na ni.

3 Implementace systému PseudoFAT

Tato sekce stručně popisuje klíčové metody implementace systému PseudoFAT. Každá metoda je uvedena s popisem svého účelu, hlavní funkcionality a výstupu.

3.1 Inicializace systému

Metoda: Konstruktor `PseudoFAT::PseudoFAT(const std::string &file)`
Účel: Zajišťuje načtení existujícího souborového systému nebo vytvoření nového při absenci souboru.
Funkce:

- Zkontroluje, zda zadaný soubor existuje.
- Pokud neexistuje, umožní uživateli provést formátování pomocí příkazu `format`.
- Pokud existuje, načte struktury systému a aktualizuje ukazatel na další volné ID.

Výstup: Úspěšné načtení nebo inicializace systému.

3.2 Formátování disku

Metoda: `PseudoFAT::formatDisk(const std::string &sizeStr)`
Účel: Inicializuje nový souborový systém s definovanou velikostí.
Funkce:

- Vypočítá a inicializuje FAT tabulky.
- Vytvoří datový soubor s nulovými hodnotami o zadané velikosti.
- Přidá kořenový adresář do struktury systému.
- Uloží stav systému do souboru.

Výstup: OK nebo chybová zpráva při selhání vytvoření souboru.

3.3 Práce s adresáři

Metoda: `PseudoFAT::createDirectory(const std::string &path)`
Účel: Vytvoří nový adresář na zadané cestě.
Funkce:

- Ověří, zda cesta obsahuje pouze jedno jméno adresáře.

- Zkontroluje, zda v cílovém adresáři již neexistuje soubor nebo adresář se stejným názvem.
- Přidá nový adresář do cílového adresáře a aktualizuje strukturu systému.

Výstup: OK nebo chybová zpráva při neplatné cestě nebo jménu.

3.4 Kontrola integrity systému

Metoda: `PseudoFAT::check()`
Účel: Zajišťuje, že FAT tabulky a datové struktury systému nejsou poškozené.
Funkce:

- Prochází FAT tabulku a kontroluje neplatné clustery (např. `FAT_BAD_CLUSTER`).
- Hledá sirotčí clustery, které nejsou propojené s žádným souborem.

Výstup: Zpráva o stavu systému – buď „No corruption detected“, nebo podrobnosti o nalezených chybách.

3.5 Mazání adresářů

Metoda: `PseudoFAT::rmdir(const std::string &path)`
Účel: Odstraní prázdný adresář na zadané cestě.
Funkce:

- Ověří, zda zadaná cesta odkazuje na adresář.
- Zkontroluje, zda je adresář prázdný.
- Odstraní adresář z rodičovské struktury a uloží změny do souboru.

Výstup: OK nebo chybová zpráva při neplatné cestě nebo neúspěšném mazání.

3.6 Přesun a kopírování souborů

Metoda: `PseudoFAT::mv(const std::string &srcPath, const std::string &destPath)`
Účel: Přesune nebo přejmenuje soubor.
Funkce:

- Ověří existenci zdrojového souboru a cílového adresáře.
- Zkontroluje konflikty názvů v cílovém adresáři.
- Aktualizuje informace o souboru a uloží změny.

Výstup: OK nebo chybová zpráva.

Metoda: `PseudoFAT::cp(const std::string &srcPath, const std::string &destPath)`
Účel: Kopíruje soubor do nového umístění.
Funkce:

- Alokuje nové clustery pro kopii souboru.
- Zkopíruje data ze zdrojových clusterů do nových.
- Přidá kopii souboru do cílového adresáře.

Výstup: OK nebo chybová zpráva.

4 Uživatelská příručka

4.1 Překlad programu

Pro překlad programu použijte následující příkazy:

```
make clean
make
```

4.2 Spuštění programu

Program se spouští pomocí:

```
./main <název_souboru>
```

Například:

```
./main fileSystem.dat
```

4.2.1 Nový souborový systém

Pokud soubor `fileSystem.dat` neexistuje, je nutné jej nejprve naformátovat. Použijte příkaz:

```
format 600MB
```

4.2.2 Otevření existujícího souborového systému

Pokud soubor již existuje a byl dříve použit, program jej načte a umožní okamžitou práci s příkazy.

4.2.3 Chyba souborového systému

Pokud soubor obsahuje chybu, například `File bad cluster`, mohou být použity pouze následující příkazy:

- `format` – Opětovné naformátování souborového systému.
- `check` – Kontrola integrity souborového systému.
- `exit` – Ukončení programu.

4.3 Dostupné příkazy

Níže je uveden seznam příkazů a jejich popis.

4.3.1 Základní příkazy

- `format <velikost>` – Formátuje souborový systém na danou velikost.
- `exit` – Ukončí program.

4.3.2 Správa adresářů

- `mkdir <cesta>` – Vytvoří nový adresář.
- `rmdir <cesta>` – Smaže prázdný adresář.
- `cd <cesta>` – Změní aktuální pracovní adresář.
- `ls <cesta>` – Vypíše obsah adresáře.

4.3.3 Práce se soubory

- `incp <zdroj> <cíl>` – Importuje soubor z pevného disku do systému.
- `outcp <zdroj> <cíl>` – Exportuje soubor ze systému na pevný disk.
- `rm <cesta>` – Smaže soubor.
- `mv <zdroj> <cíl>` – Přesune nebo přejmenuje soubor/adresář.
- `cp <zdroj> <cíl>` – Zkopíruje soubor/adresář.

4.3.4 Další příkazy

- `cat <cesta>` – Vypíše obsah souboru.
- `info <cesta>` – Zobrazí informace o souboru nebo adresáři.

- `load <soubor>` – Načte příkazy ze souboru a provede je.
- `check` – Zkontroluje integritu souborového systému.
- `bug <cesta>` – Poškodí souborový systém pro testování.

4.4 Chybová hlášení

- `INVALID PATH` – Neplatná cesta.
- `PATH NOT FOUND` – Cesta nebyla nalezena.
- `FILE NOT FOUND` – Soubor nebyl nalezen.
- `SAME NAME` – Název již existuje.
- `NOT ENOUGH SPACE` – Nedostatek místa pro operaci.

5 Závěr

Popisované metody demonstrují klíčové části implementace systému PseudoFAT. Každá metoda je navržena tak, aby splňovala požadavky na správu souborového systému a zároveň zajišťovala robustní detekci a opravu chyb.