FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z KIV/ZOS

Souborový systém založený na FAT

Petr Vondrovic
A20B0273P
vondrp@students.zcu.cz

20. prosince 2022

Zadání

Implementujte zjednodušený souborový systém založený na pseudo FAT. Při implementaci stačí použít jedinou FAT tabulku a předpokládejte, že adresář se vždy vejde do jednoho clusteru.
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`
   `Možný výsledek:`
   `OK`
   `FILE NOT FOUND (není zdroj)`
   `PATH NOT FOUND (neexistuje cílová cesta)`

2. Přesune soubor s1 do umístění s2, nebo přejmenuje s1 na 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) EXISTS (nelze založit, již existuje)`

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

6. Vypíše obsah adresáře a1, bez parametru vypíše obsah aktuálního adresáře
   `ls a1`
   `ls`
   `Možný výsledek:`
   `OK`
   `FILE: f1 DIR: a2 PATH NOT FOUND (neexistující adresář)`

7. Vypíše obsah souboru s1
   `cat a1`
   `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:`
    `S1 2,3,4,7,10`
    `FILE NOT FOUND (není zdroj)`

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

12. Nahraje soubor s1 z vašeho FS 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 spuštění 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. Defragmentace souboru - Zajistí, že datové bloky souboru s1 budou ve filesystému uložené za sebou, což si můžeme ověřit příkazem info. Předpokládáme, že v systému je dostatek místa, aby nebyla potřeba přesouvat datové bloky jiných souborů
    `defrag s1`
    `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).
Maximální délka názvu souboru bude 8+3=11 znaků (jméno.přípona) + \0 (ukončovací znak v C/C++), tedy 12 bytů.

Struktura souborového systému (fs)

```
1 struct boot_record {
2    int32_t disk_size;
3    int32_t cluster_size;
4    int32_t cluster_count;
5    int32_t fat1_start_address;
6    int32_t data_start_address;
7 };
```

Zdrojový kód 1: Super block fs

```
1 struct directory_item {
2    char name[13];
3    int32_t isFile;
4    int32_t size;
5    int32_t start_cluster;
6 };
```

Zdrojový kód 2: Struktura adresáře

Souborový systém je v souboru v němž se nachází následující položky:

1. Super block - viz 1, obsahuje základní informace o datech fs
2. FAT tabulka: tabulka položek popisující obsah clusterů souboru. Může obsahovat:
    - odkaz na další položku bloku
    - volný block
    - konec souboru
    - špatný/poškozený blok
3. Root directory: struktura 2, sloužící jako popis kořenové položky fs
4. Clustery - data uložená ve fs. Na pozici dat ukazují položky FAT tabulky
   Data clusterů jsou dva typy:

- adresář: obsahuje pouze pouze položky adresáře viz 2, popisující adresář/soubor na jehož počáteční pozici ve FAT tabulce ukazují.
- data souboru: data souboru uloženého v souborovém systému

Popis adresářů a souborů

Adresář fat

Adresář fat obsahuje soubor `fat.c` a jeho hlavičku `fat.h`. Tyto soubory obsahují deklaraci struktur 1 a 2 a funkce jež s nimi, FAT tabulkou a se souborem fs pracují.
Mezi takové funkce patří např. equals, inicializace 1 a 2, nalezení volného clusteru, zapsaní 2, aktualizace FAT tabulky v souboru, odstranění 2, zjištění zda soubor/adresář existuje v adresáři `directory_item` 2 a další.

Adresář input

Tato složka obsahuje soubory `checkInput` a `inputHandler`, které mají na starosti kontrolu a zpracování vstupu od uživatele.
Nejdůležitější funkce `checkInput` jsou:

- process_path - zpracuje daný řetězec cesty na cestu vycházející z `ROOT` `directory_item`
- split_path - rozdělení dané cesty na jednotlivé části (dělič cesty je „/“)

Další funkce `checkInput` jsou například: získání jména (poslední části) z cesty, kontrola délky jména (maximum je 12 znaků), zkrácení cesty o její poslední.
Mezi hlavní funkce `inputHandler` patří:

- process_input - funkce zpracovávající vstup uživatele do konzole
- call_commands - zavolání příkazu

Dále se zde nachází funkce pro získání řádky textu a další funkce pro její rozdělení na jednotlivá slova.

Adresář output

Adresář `output` se stará o výstup příkazů. Nachází se zde `errors.h`, kde jsou napsané všechny možné výstupy příkazů. Ty slouží jako parametr funkce `print_error_message` ze souboru `messages.c/h`, která vypíše do konzole výsledek příkazu.

Adresář commands

V složce `commands` se nachází soubory, obsahující implementace příkazů souborového systému a jejich pomocné metody. Příkazy jsou popsané v zadání.

- dirCommands - příkazy pracující s adresáři fs
  příkazy: `mkdir`, `rmdir`, `ls`

- fileCommands - příkazy pracující se soubory, v některých případech jsou použitelné i pro adresáře
  příkazy: `cp`, `rm`, `mv`, `cat`, `load`, `incp`, `outcp`, `info`, `defrag`.
- pathCommands - příkazy pracující s aktuální cestou (kde se v souborovém systému nacházíme)
  příkazy: `pwd`, `cd`
- fsCommands - obsahuje pouze příkaz `format` sloužící k formátování souborového systému a pomocné metody. Mezi pomocné metody patří:
  - process_units - zpracuje parametr velikosti fs
  - format_fs - zformátování/vytvoření nového fs
  - open_fs, close_fs - načtení fs a jeho zavření

Poznámky k příkazům

Při přesunutí souboru příkazem `mv` dojde pouze k přemístění `directory_item` (2) a pozice FAT tabulky zůstanou stejné. To se liší od příkazu `cp`, při nemž dojde k nakopírování dat na nové pozice ve FAT tabulce.
U příkazu `cp` je nutné uvést jméno kopie.
U příkazu `incp` není nutné spolu s místem umístěním na souborový systém úvést název souboru nebude-li uveden bude soubor pojmenován stejně jako na vnějším systému. Naopak u příkazu `outcp` je nutné jméno souboru uvést. U `incp` je nutné uvést kam se má soubor vložit, chceme-li soubor vložit do aktuální složky použijeme cestu tečky „.“.
Příkaz `info` kromě seznamu clusterů, kde se soubor nachází, vypíše i jméno souboru, velikost a informace zda se jedná o soubor nebo adresář.
Při formátování souboru, lze uvést tři typy jednotek - kB, MB a GB. Pro zabránění přetečení datového typu `int32_t` užívaného v superblocku (1) mají jednotky nastavené maximální velikost - 2097151kB, 2047MB a 1GB. U kB je navíc stanovena minimální velikost na 2kB, protože velikost jednoho clusteru je 512B a při menší velikosti by nebylo dost místa ani na cluster pro kořen souborového systému.

main.c

Soubor `main.c` je spouštěcí/hlavní soubor aplikace.

Popis principu implementace příkazů

Při impelementaci příkazů se užívá několika globálních proměnných.

- ukazatel na soubor souborového systému
- jméno souborového systému
- ukazatel na super block 1
- ukazatel na FAT tabulku

- ukazatel na kořenový (root) adresář
- ukazatel na adresář v němž se uživatel zrovna nachází
- současná cesta
- maximální délka cesty

Při implementaci příkazů se vždy nejprve zpracuje cesta k souboru/adresáři pomocí `process_path` a na konci většiny příkazu se zavolá `print_error_message` k informování o úspěchu/neúspěchu provedení příkazu.
K nalezení příslušných `directory_item` 2 se používá použití cesty k nim. K tomu je třeba znát `directory_item` z něhož hledání začíná, z toho důvodu `process_path` upraví cestu, aby šla od kořene (root). Cesty se také používá k nalezení rodiče a prarodiče `directory_item`, které je nutné znát, pro přepsání změn v souboru souborového systému.

Uživatelská dokumentace

Překlad

Pro překlad programu je potřeba nástroj `cmake`, `make` a překladač `gcc`. Nejprve vygenerujeme soubor makefile zadáním příkazu `cmake -B ./build/`. Přejdeme do adresáře `build` a v něm spustíme překlad programu příkazem `make`. Po dokončení operace by měl být vytvořen v adresáři `build` soubor `zos_sp`.

Spuštění a běh programu

Program se s parametrem názvu souborového systému. Druhý možný parametr je velikost ve formátu <velikost><jednotka>, kde jednotka může být `kB`, `MB` nebo `GB`.

Po úspěšném spuštění programu může uživatel psát příkazy. Pro ukončení běhu programu je třeba napsat příkaz `end`.

Závěr

Souborový systém alokuje data pomocí FAT tabulku a má naimplementované všechny příkazy ze zadání. Program běží stabilně, nepadá a všechny příkazy jsou ošetřené. Požadavky na práci jsou splněny.