FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská univerzita

Defragmentace pseudoFAT tabulky

Semestrální práce - UPS

Petr Štechmüller

1. ledna 2017

Obsah

Zadání **2**

Implementace **3**
Vytvoření nového souboru 4
Smazání souboru 5
Výpis clusterů obsahující data souboru 5
Výpis obsahu souboru 5
Vytvoření prázdné složky 6
Smazání prázdné složky 6
Výpis stromové struktury 6
Defragmentace 7

Závěr **8**

Zadání

Vytvořte jednoduchý souborový systém, který bude organizován pomocí pseudoFAT tabulky. Nad souborovým systémem budete provádět následující operace:

- Vytvoření nového souboru
- Smazání souboru
- Vypsat čísla clusterů obsahující data zadaného souboru
- Vypsat obsah souboru
- Vytvoření prázdné složky
- Smazání prázdné sožky
- Vypsat stromovou strukturu celého souborového systému
- Provést úplnou defragmentaci souborového systému

Implementace

Souborový systém se snaží simulovat prostředí v linuxu. Jako znak oddělující jednotlivé soubory a složky od sebe je použito "/". Kořenový adresář se jmenuje "/". Při implementaci byla použity následující struktury pro boot record1 a root directory2:

```
struct boot_record {
    char volume_descriptor[251];
    int fat_type;
    int fat_copies;
    unsigned int cluster_size;
    unsigned long root_directory_max_entries_count;
    unsigned int cluster_count;
    unsigned int reserved_cluster_count;
    char signature[4];
};
```

Listing 1: Struktura pro boot record

```
struct root_directory {
    char file_name[13];
    char file_mod[10];
    short file_type;
    long file_size;
    unsigned int first_cluster;
};
```

Listing 2: Struktura pro root directory

Celý souborový systém je implementován v jazyce C++ a je reprezentován třídou Fat. Třída Fat obsahuje následující veřejné metody3:

```
void loadFat();

std::shared_ptr<root_directory> findFileDescriptor(const std::↩
    string &t_path);

const std::vector<unsigned int> getClusters(std::shared_ptr<↩
    root_directory> t_fileEntry);

```

```
7  void tree();
8
9  void createEmptyFat();
10
11 void createDirectory(const std::string &t_path, const std::string↩
       &t_addr);
12
13 void deleteDirectory(const std::string &t_pseudoPath);
14
15 void insertFile(const std::string &t_filePath, const std::string ↩
     &t_pseudoPath);
16
17 void deleteFile(const std::string &t_pseudoPath);
18
19 const std::string getClusterContent(const unsigned int t_index);
20
21 const std::vector<std::string> readFileContent(const std::↩
     shared_ptr<root_directory> t_rootDirectory);
22
23 const std::vector<std::string> readFileContent(const unsigned int↩
       t_index, const long t_fileSize);
24
25 void writeClusterContent(const unsigned int t_index, const std::↩
     string &t_data);
26
27 void save();
28
29 void printBootRecord();
30
31 void printRootDirectories();
32
33 void printRootDirectory(const std::shared_ptr<root_directory> ↩
     t_rootDirectory);
34
35 void printClustersContent();
36
37 void printFileContent(const std::vector<std::string> &↩
     t_fileContent);
38
39 void printSubTree(const std::shared_ptr<root_directory> ↩
     t_rootDirectory, const unsigned int t_depth);
```

Listing 3: Veřejné metody třídy Fat

Vytvoření nového souboru

Při vytváření nového souboru se nejdříve zkontroluje, jestli existuje kopírovaný soubor. Dále se najde rodičovský adresář4 nového souboru.

```
1 auto separatorIndex = t_pseudoPath.find_last_of(PATH_SEPARATOR);
2 auto realPseudoPath = t_pseudoPath.substr(0, separatorIndex);
3 auto parentDirectory = findFileDescriptor(realPseudoPath);
4 auto parentDirectoryContent = loadDirectory(parentDirectory->↩
    first_cluster);
```

```
5  auto fileName = t_pseudoPath.substr(separatorIndex + 1);
6
7  std::fseek(workingFile, 0L, SEEK_END);
8
9  auto fileSize = std::ftell(workingFile);
10 auto file = makeFile(fileName, "rwxrwxrwx", fileSize, ↩
       FILE_TYPE_FILE, getFreeCluster());
```

Listing 4: Inicializace proměnných pro vložení nového souboru

Dále se zkontroluje, zda-li není rodičovská složka plná a neobsahuje soubor stejného názvu. Když projdou všechny validace, tak se najdou volné clustery pro soubor a uloží se obsah. Nakonec se uloží záznam v rodičovské složce o novém souboru. Příklad použití5

```
1 ./zos_semestralka_fat test.fat -a ./f.txt /f.txt
```

Listing 5: Vytvoření nového souboru

Smazání souboru

Před samotným smazáním souboru se načte obsah rodičovské složky a odstraní se záznam o souboru. Dále se vymažou záznamy ve fat tabulce. Obsah v clustrech zůstane zachován.

```
1 ./zos_semestralka_fat test.fat -f /f.txt
```

Listing 6: Smazání souboru

Výpis clusterů obsahující data souboru

Příklad výpisu clusterů7

```
1 ./zos_semestralka_fat test.fat -c /a.txt
2
3 /a.txt: 1, 2, 3, 4,
```

Listing 7: Výpis clusterů obsahující data souboru

Výpis obsahu souboru

Příklad výpisu clusterů8

```
1 ./zos_semestralka_fat test.fat -l /a.txt
```

```
2
3 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris ↩
    commodo, ipsum ac commodo imperdiet, tortor ante euismod nisi↩
    , quis aliquet nibh quam eu justo. Lorem ipsum dolor sit amet↩
    , consectetur adipiscing elit. Sed in feugiat lacus. Morbi ↩
    eget ante varius, tincidunt neque consequat, tincidunt tellus↩
    . Cras hendrerit mi vitae convallis mollis. Donec eu nisi ↩
    vehicula, pellentesque sem at, interdum sapien. Sed ↩
    ullamcorper tristique sapien, in rhoncus metus hendrerit non.↩
     Nam elementum neque enim, vitae laoreet ligula blandit nec.
```

Listing 8: Výpis obsahu zadaného souboru

Vytvoření prázdné složky

Vytvoření složky9 probíhá velmi podobně jako vytvoření nového souboru.

```
1 ./zos_semestralka_fat test.fat -m bin /
```

Listing 9: Smazání souboru

Smazání prázdné složky

Smazání složky10 probíhá stejným způsobem, jako smazání souboru s tím rozdílem, že před smazáním se zkontroluje, zda-li je složka prázdná. Pokud složka prázdná není, tak se nesmaže.

```
1 ./zos_semestralka_fat test.fat -r /bin
```

Listing 10: Smazání souboru

Výpis stromové struktury

Pro výpis stromové struktury11 se používá rekurze. Když se narazí na složku, tak se zavolá rekurzivně metoda pro výpis stromové struktury.

```
1 ./zos_semestralka fat test.fat -p
2 +/
3   +bin 1 (1)
4       -a.txt 11 (1)
5       -b.txt 12 (1)
6       -c.txt 13 (1)
7   +home 2 (1)
8       +petr 6 (1)
9           -b.txt 7 (1)
10          -c.txt 8 (1)
```

```
11        +michal 9 (1)
12        +filip 10 (1)
13    -a.txt 3,4,5 (3)
```

Listing 11: Výpis stromové struktury

Defragmentace

Úplná defragmentace spočívá v tom, že se nejenom odstraní mezery mezi soubory, ale také se linearizují obsahy souborů tak, aby clustery jednoho souboru byly pohromadě. Při defragmentaci se přesouvají pouze clustery souborů. Pokud se narazí na cluster, který obsahuje složku, tak je přeskočen. Celý proces defragmentace běží pouze v jednom vlákně, protože paralelizace neměla smysl. Pouze načtení kompletní stromové struktury před defragmentací se provádí paralelně. Pro dosažení paralelismu se používá Threadpool. Defragmentaci je nutné spouštět pomocí přiloženého skriptu **defragmenter.sh**12. V průběhu defragmentace se vypisuje průběh procesu.

```
1 ./defragmenter.sh test.fat
2
3 Presouvam cluster 19 na novou pozici: 8
4 Presouvam cluster 20 na novou pozici: 9
5 Presouvam cluster 131 na novou pozici: 23
6 Presouvam cluster 132 na novou pozici: 24
7 Presouvam cluster 133 na novou pozici: 25
```

Listing 12: Spuštění defragmentace

Závěr

Vytvořit vlastní souborový systém byla velká výzva. Základní práce jako vytváření a odstraňování souborů a složek byla jednoduchá, defragmentace byla složitější. Vzhledem k tomu, že paralelně se pouze načítá stromová struktura a zbytek se provádí v jednom vlákně, tak nejsou přiloženy žádné srovnávací časy.