



ZÁPADOČESKÁ UNIVERZITA

DEFRAGMENTACE PSEUDOFAT TABULKY

**Semestrální práce - UPS**

*Petr Štechmüller*

1. ledna 2017

# Obsah

<b>Zadání</b>	<b>2</b>
<b>Implementace</b>	<b>3</b>
Vytvoření nového souboru . . . . .	4
Smazání souboru . . . . .	5
Výpis clusterů obsahující data souboru . . . . .	5
Výpis obsahu souboru . . . . .	5
Vytvoření prázdné složky . . . . .	5
Smazání prázdné složky . . . . .	6
Výpis stromové struktury . . . . .	6
Defragmentace . . . . .	6
<b>Závěr</b>	<b>7</b>

# Zadání

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

- Vytvoření nového souboru
- Smazání souboru
- Vypsání čísla clusterů obsahující data zadaného souboru
- Vypsání obsahu souboru
- Vytvoření prázdné složky
- Smazání prázdné složky
- Vypsání 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:

```
1 struct boot_record {
2     char volume_descriptor[251];
3     int fat_type;
4     int fat_copies;
5     unsigned int cluster_size;
6     unsigned long root_directory_max_entries_count;
7     unsigned int cluster_count;
8     unsigned int reserved_cluster_count;
9     char signature[4];
10 };
```

Listing 1: Struktura pro boot record

```
1 struct root_directory {
2     char file_name[13];
3     char file_mod[10];
4     short file_type;
5     long file_size;
6     unsigned int first_cluster;
7 };
```

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:

```
1 void loadFat();
2
3 std::shared_ptr<root_directory> findFileDescriptor(const std::string &t_path);
4
5 const std::vector<unsigned int> getClusters(std::shared_ptr<root_directory> t_fileEntry);
6
```

```

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 adresář 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 použití<sup>5</sup>

```

1 ./zos_semestralka fat nazev_fatky -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 nazev_fatky -f /f.txt

```

Listing 6: Smazání souboru

## Výpis clusterů obsahujících data souboru

Příklad výpisu clusterů<sup>7</sup>

```

1 ./zos_semestralka fat nazev_fatky -c /f.txt
2
3 /f.txt: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ↵
    17, 18, 19, 20, 21, 22,

```

Listing 7: Výpis clusterů obsahujících data souboru

## Výpis obsahu souboru

### Vytvoření prázdné složky

Vytvoření složky probíhá velmi podobně jako vytvoření nového souboru.

## Smazání prázdné složky

Smazání složky 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.

## Výpis stromové struktury

Pro výpis stromové struktury se používá rekurze. Když se narazí na složku, tak se zavolá rekurzivně metoda pro 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 Threadpoolu. Defragmentaci je nutné spouštět pomocí přiloženého skriptu **defragmenter.sh**.

# Závěr

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