FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce z předmětu KIV/ZOS

Virtuální souborový systém na bázi NTFS

Patrik Harag

harag@students.zcu.cz

(A15B0034P)

2. ledna 2018

Obsah

Kapitola 1

Úvod

1.1 Zadání

Implementace virtuálního souborového systému na bázi NTFS s podporou následujících příkazů:

- `pwd` – vypíše aktuální cestu.
- `cd <path>` – změní aktuální cestu.
- `cp <src_path> <dest_path>` – zkopíruje soubor.
- `mv <src_path> <dest_path>` – přesune soubor nebo složku.
- `rm <path>` – smaže soubor.
- `mkdir <path>` – vytvoří adresář.
- `rmdir <path>` – smaže prázdný adresář.
- `ls [path]` – vypíše obsah adresáře.
- `cat <path>` – vypíše obsah souboru.
- `info <path>` – vypíše informace o souboru/adresáři
- `incp <src_path> <dest_path>` – nahraje soubor z virtuálního *FS* do hostitelského *FS*.
- `outcp <src_path> <dest_path>` – nahraje soubor z hostitelského *FS* do virtuálního *FS*.
- `load <path>` – načte soubor s příkazy a sekvenčně je vykoná.

Dále implementace defragmentace a kontroly konzistence.

Více v přiloženém zadání.

Kapitola 2

Programátorská dokumentace

Program byl vyvinut v C 99.

2.1 Seznam zdrojových souborů

- *main.c* – obsahuje vstupní funkci, zpracovává parametry a spouští REPL.
- *utils.c/h* – obsahuje různé pomocné funkce.
- *repl.c/h* – obsahuje *read–eval–print-loop* (CLI).
- *commands.c/h* – obsahuje obsluhu všech příkazů, především ošetření vstupů.
- *io.c/h* – řeší zápis/čtení pseudo NTFS do/ze souboru.
- *vfs.c/h* – obsahuje základní funkce pro práci s *VFS* (Virtual File System).
- *vfs-bitmap.c/h* – obsahuje nízkoúrovňové funkce pro práci s bitmapou.
- *vfs-boot-record.c/h* – obsahuje nízkoúrovňové funkce pro práci s *boot record*.
- *vfs-mft.c/h* – obsahuje nízkoúrovňové funkce pro práci s *MFT*.
- *vfs-module-allocation.c/h* – obsahuje funkce, které umožňují alokovat a uvolňovat různé zdroje.
- *vfs-module-consistency.c/h* – obsahuje funkce pro ověřování konzistence *VFS*.
- *vfs-module-defragmentation.c/h* – obsahuje funkce pro defragmentaci.
- *vfs-module-find.c/h* – obsahuje funkce pro vyhledávání *MFT* položek.
- *vfs-module-utils.c/h* – obsahuje různé pomocné funkce pro práci s *VFS*.
- *vfs-module-validation.c/h* – obsahuje různé validační funkce.

2.2 Důležité datové struktury

VFS

Tato datová struktura, která se nachází v souboru *vfs.h*, uchovává všechny důležité informace o *VFS*.

```
typedef struct _VFS {
    char* file_name;
    FILE* file;
    bool initialized;

    BootRecord* boot_record;
    Mft* mft;
    Bitmap* bitmap;
} VFS;
```

BootRecord

Struktura ze souboru *vfs-boot-record.h*, která definuje *boot record* tak, jak se bude ukládat do souboru.

```
typedef struct _BootRecord {
    char signature[16];           // login autora FS
    char volume_descriptor[48];   // popis vygenerovaného FS
    int32_t disk_size;            // maximální velikost obsahu
    int32_t cluster_size;         // velikost clusteru
    int32_t cluster_count;        // pocet clusteru
    int32_t mft_item_count;       // počet položek v mft
    int32_t bitmap_size;          // velikost bitmapy v bytech
} BootRecord;
```

Bitmap

Struktura ze souboru *vfs-bitmap.h*, která definuje strukturu ukládající obsah bitmapy.

```
typedef struct _Bitmap {
    int32_t length;
    unsigned char* data;
    int32_t cluster_count;
} Bitmap;
```

Mft, MftItem, MftFragment

V souboru *vfs-mft.h* jsou definovány struktury pro práci s *MFT*.

```
/** Struktura fragmentu MFT položky tak, jak se bude ukládat do souboru */
typedef struct _MftFragment {
    int32_t start_index;   // start adresa
    int32_t count;         // pocet clusteru ve fragmentu
} MftFragment;

/** Struktura MFT položky tak, jak se bude ukládat do souboru */
typedef struct _MftItem {
    int32_t uid;           // UID polozky, pokud UID = UID_ITEM_FREE, je
       polozka volna
    int32_t parent_uid;
    int8_t is_directory;   // soubor, nebo adresar
    int8_t item_order;     // poradi v MFT pri vice souborech, jinak 1
    int8_t item_total;     // celkovy pocet polozek v MFT
    char item_name[VFS_MFT_ITEM_NAME_LENGTH];
    int32_t item_size;     // velikost souboru v bytech
    MftFragment fragments[VFS_MFT_FRAGMENTS_COUNT]; //fragmenty souboru
} MftItem;

/** Pomocná struktura pro MFT */
typedef struct _MFT {
    int32_t length;
    MftItem* data; // pole struktur
} Mft;
```

2.3 Defragmentace

Defragmentace může být vyvolána příkazem `defragment`.

Je implementována úplná defragmentace disku. Po provedení úplné defragmentace se každý soubor skládá z maximálně jednoho fragmentu a všechny využité clustery jsou umístěny na začátek disku.

Implementace

Algoritmus funguje tak, že postupně, od prvního, procházíme clustery a na dané místo se snažíme umístit obsah tak, aby po dokončení tohoto cyklu byl disk defragmentován.

Při zpracování dalšího clusteru vždy najdeme první následující neprázdný cluster a podle něj určíme MFT položku (pomocí předpřipravené tabulky), která jej využívá. Dalším cílem bude tuto MFT položku, tedy jejích n clusterů umístit na daný cluster a případně n - *1* následujících clusterů.

Ve vnitřním cyklu postupně kopírujeme clustery a v případě, že některý cílový cluster není volný, realokujeme celý soubor, kterému patří někam pryč (pokud je

dostatek místa, tak někam dál, abychom jej hned v příští iteraci nemuseli opět realokovat). Po skončení vnitřního cyklu pokračujeme za posledním umístěným clusterem.

2.4 Kontrola konzistence

Kontrola konzistence je prováděna při každém načtení virtuálního souborového systému ze souboru, případně může být vyvolána příkazem `check consistency`.

Jsou prováděny dvě kontroly:

- Zda velikosti položek odpovídají počtu jejich clusterů.
- Zda není cluster referencován více položkami.

Implementace

Úloha je paralelizována. Je vytvořeno n vláken a každé vlákno si říká o další položky ke kontrole. Nad položkou provede obě dvě kontroly.

Je přitom využívána sdílená struktura společná pro všechny vlákna:

```
typedef struct _CheckContext {
    VFS* vfs;
    int32_t i; // index další položky ke zpracování
    bool consistent;
    pthread_mutex_t mutex;
    MftItem** clusters; // tabulka cluster index => MftItem*
} CheckContext;
```

Kapitola 3

Uživatelská dokumentace

Sestavení Pro sestavení je vyžadován GNU Linux. Sestavení proběhne po zadání příkazu `make` v kořenovém adresáři.

Spuštění Pro spuštění je vyžadován GNU Linux. Spuštění proběhne po zadání příkazu `./build/ntfs-dist <název souboru>`.

3.1 Seznam příkazů

Inicializační příkazy

- `init` – inicializuje *VFS* o velikosti disku 10 kB (jako `init 10240`). Velikost clusteru je 256 B.
- `init <disk_size_in_bytes>` – inicializuje *VFS*. K zadané velkosti se připočte ještě *boot record*, bitmapa a *MFT* (které bude mít cca 10% velikosti disku). Velikost clusteru je 256 B.
- `init <cluster_count> <mft_items_count>` – inicializuje *VFS*. Velikost clusteru je 256 B.
- `init <cluster_count> <cluster_size_in_bytes> <mft_items_count>` – inicializuje *VFS*.

Informativní a debug příkazy

- `df` – vypíše informace o obsazenosti.
- `show parameters` – vypíše parametry *VFS*.
- `show bitmap` – zbrazí bitmapu (0 → volný cluster, 1 → obsazený cluster).
- `show mft` – vypíše položky *MFT*.

Základní příkazy

- `pwd` – vypíše aktuální cestu.
- `cd <path>` – změní aktuální cestu.
- `cp <src_path> <dest_path>` – zkopíruje soubor.
- `mv <src_path> <dest_path>` – přesune soubor nebo složku.
- `rm <path>` – smaže soubor.
- `mkdir <path>` – vytvoří adresář.
- `rmdir <path>` – smaže prázdný adresář.
- `ls [path]` – vypíše obsah adresáře.
- `cat <path>` – vypíše obsah souboru.
- `info <path>` – vypíše informace o souboru/adresáři
- `incp <src_path> <dest_path>` – nahraje soubor z virtuálního *FS* do hostitelského *FS*.
- `outcp <src_path> <dest_path>` – nahraje soubor z hostitelského *FS* do virtuálního *FS*.
- `load <path>` – načte soubor s příkazy a sekvenčně je vykoná.

Ostatní příkazy

- `reallocate <path>` – realokuje soubor na jinou pozici - to ho může defragmentovat, pokud je k dispozici vhodné místo.
- `defragment` – provede úplnou defragmentaci disku.
- `check consistency` – zkontroluje konzistenci (kontrola konzistence se automaticky spouští po načtení).
- `disable first-fit` – deaktivuje metodu pro vyhledávání volného místa na disku *first fit*. Po deaktivaci budou vybírány vždy první volné clustery, bude tak docházet k větší fragmentaci. Slouží pro testovací účely.

Kapitola 4

Závěr

Implementoval jsem virtuální souborový systém na bázi NTFS podle zadání. Jsou podporovány všechny požadované příkazy, defragmentace a kontrola konzistence. Byly také vytvořeno několik příkazů navíc, většina z nich pro účely kontroly správného fungování programu.

Během celého vývoje jsem vytvářel funkční testy, což se mi velmi vyplatilo především v pozdějších fázích vývoje, kdy už by byl problém „ohlídat“ takové množství funkcí a vlastností. Zvláště v jazyce, jako je ANSI C.