

Semestrální práce z KIV/ZOS

Souborový systém pseudoNTFS

Zdeněk Častorál
A16B0467P
zcastora@students.zcu.cz

4. 2. 2019

Obsah

1	Zadání	3
2	Programátorská dokumentace	6
2.1	Struktura NTFS	6
2.2	Realizace programu	7
2.2.1	Moduly programu	7
2.2.2	Běh programu	8
2.2.3	Příkazy programu	8
2.2.4	Překlad programu	9
3	Uživatelská dokumentace	10
3.1	Spuštění programu	10
3.2	Ovládání programu	10
4	Závěr	12
4.1	Testování	12
4.2	Zhodnocení práce	12

1 Zadání

Tématem semestrální práce je návrh a implementace souborového systému pseudoNTFS. Vytvořený souborový systém bude umožňovat vykonávání základních příkazů nad soubory a adresáři. Jedná se například o vytvoření adresáře, kopírování souborů, přesouvání souborů atd.

Program se bude spouštět s jedním parametrem a tím bude název souborového systému (např. `myFS`). Po spuštění bude program čekat na zadání jednotlivých příkazů (všechny soubory mohou být zadány jak absolutní, tak relativní cestou).

Práce bude realizována v jazyce *C/C++*.

Jednotlivé příkazy nad souborovým systémem jsou uvedeny níže.

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`.

```
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)

EXIST (nelze založit, již existuje)

5. Smaže prázdný adresář `a1`.

```
rmdir a1
```

Možný výsledek:

OK

FILE NOT FOUND (neexistující adresář)

NOT EMPTY (adresář obsahuje podadresáře, nebo soubory)

6. Vypíše obsah adresáře a1.

```
ls a1
```

Možný výsledek:

-FILE

+DIRECTORY

PATH NOT FOUND (neexistující adresář)

7. Vypíše obsah souboru s1.

```
cat s1
```

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 fragmentech/clusterech se nachází), uid, ...

```
info a1/s1
```

Možný výsledek:

NAME - UID - SIZE - FRAGMENTY - CLUSTERY

FILE NOT FOUND (není zdroj)

11. Nahraje soubor s1 z pevného disku do umístění s2 v pseudoNTFS.

```
incp s1 s2
```

Možný výsledek:

OK

FILE NOT FOUND (není zdroj)
PATH NOT FOUND (neexistuje cílová cesta)

12. Nahraje soubor s1 z pseudoNTFS 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. Vytvoří symbolický link na soubor s1 s názvem s2. Dále se s ním pracuje očekávaným způsobem, tedy např. cat s2 vypíše obsah souboru s1.

```
slink s1 s2
```

Možný výsledek:

OK

FILE NOT FOUND (není zdroj)

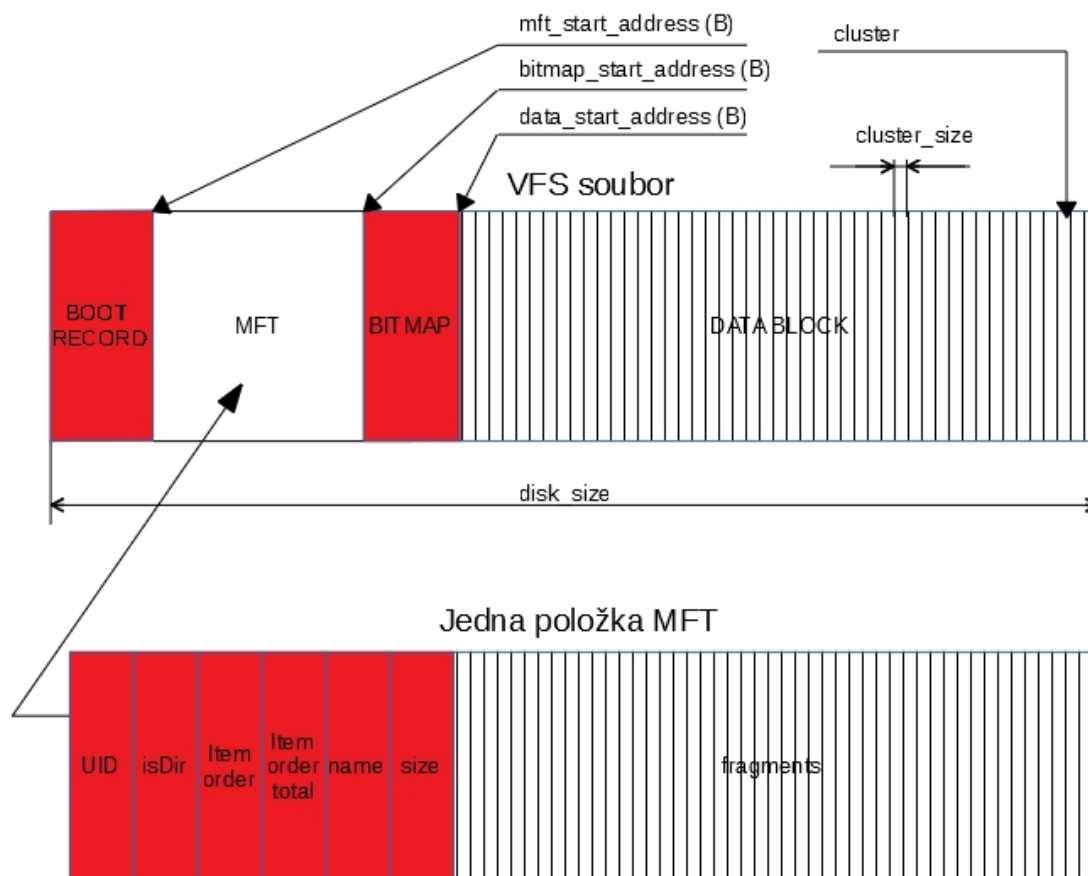
PATH NOT FOUND (neexistuje cílová cesta)

Budeme předpokládat korektní zadání syntaxe příkazů, nikoliv však sémantiky (tj. např. cp s1 zadáno nebude, ale může být zadáno cat s1, kde s1 neexistuje).

2 Programátorská dokumentace

2.1 Struktura NTFS

Souborový systém NTFS se skládá ze 4 hlavních částí (viz obrázek 2.1).



Obrázek 2.1: Struktura souborového systému NTFS

Části souborového systému NTFS jsou:

- **boot record:**
 - uchovává informace o celém souborovém systému, např. název, popis, velikost, adresa počátku MFT záznamů, adresa počátku bitmapy atd.,
 - je umístěn na začátku oblasti vymezené pro souborový systém,
- **MFT záznamy:**
 - spravují informace o souborech,
 - každý záznam obsahuje: unikátní číslo, typ položky: soubor/adresář/..., název souboru, velikost souboru, pořadí záznamu, celkový počet záznamů, popis fragmentů,
 - MFT záznamy jsou umístěny za boot recordem,

- **bitmapa:**
 - spravuje informace datové části – které bloky jsou volné a které obsazené,
 - velikost bitmapy se rovná počtu clusterů datové části souborového systému,
 - bitmapa je umístěna za MFT záznamy,
- **datová část:**
 - v datové části jsou fyzicky uložena data souborů,
 - je rozdělena na clustery pevné velikosti,
 - datová část je umístěna za bitmapou a je tak poslední částí souborového systému.

2.2 Realizace programu

Program je dle zadání implementován v jazyce C a odladěn pro fungování v prostředí operačních systémů *Windows*. Konkrétně byl program vyvíjen a laděn na systému *Windows 10 Enterprise*, k překladu byl použit překladač `gcc 4.9.2`.

Aplikace je rozdělena do několika modulů, které jsou blíže popsány v kapitole 2.2.1.

2.2.1 Moduly programu

Program je rozdělen do devíti modulů a osmi hlavičkových souborů, které jednotlivým modulům přísluší.

- Modul `main.c` (nemá svůj hlavičkový soubor):
 - je hlavním modulem programu,
 - sjednocuje všechny komponenty programu (moduly a hlavičkové soubory) a obsahuje spustitelný bod programu,
 - zpracovává argumenty příkazové řádky,
 - připravuje program pro ukončení (zavření proudů, uvolnění paměti atd.).
- Modul `shell.c` (jeho hlavičkový soubor – `shell.h`):
 - reprezentuje shell, který zpracovává vstup od uživatele,
 - žádá ostatní komponenty programu o zpracování požadavku.
- Modul `command.c` (jeho hlavičkový soubor – `command.h`):
 - slouží k rozdělení vstupu od uživatele na *příkaz* a *parametry*,
 - také slouží jako přepravka těchto hodnot.
- Modul `process_command.c` (jeho hlavičkový soubor – `process_command.h`):
 - slouží ke zpracování jednotlivých příkazů,
 - volá příslušné funkce pro vykonání požadované funkcionality zadaného příkazu.
- Modul `functions.c` (jeho hlavičkový soubor – `functions.h`):
 - obsahuje funkce pro vykonávání jednotlivých funkcí shellu,

- funkce tohoto modulu jsou volány modulem `process_command.c`,
- funkce tohoto modulu využívají funkce modulu `functions_helper.c`.
- Modul `functions_helper.c` (jeho hlavičkový soubor – `functions_helper.h`):
 - obsahuje pomocné funkce pro realizaci funkcí pro vykonávání jednotlivých příkazů shellu,
 - funkce tohoto modulu jsou volány modulem `process_command.c` a `functions.c`.
- Modul `file_manager.c` (jeho hlavičkový soubor – `file_manager.h`):
 - stará se o práci s datovým souborem pro uložení souborového systému,
 - stará se o jeho vytvoření, uložení struktur, formátování atd.
- Modul `fs_structures.c` (jeho hlavičkový soubor – `fs_structures.h`):
 - vytváří struktury potřebné pro souborový systém,
 - obsahuje funkce pro základní operace s těmito strukturami.
- Modul `global_vars.c` (jeho hlavičkový soubor – `global_vars.h`):
 - reprezentuje přepravku pro globální proměnné celé aplikace.

2.2.2 Běh programu

Program se spouští s jedním parametrem, který obsahuje název souborového systému – například `myFS`.

Při prvním spuštění tento soubor zatím neexistuje, je nutné jej vytvořit a zformátovat. Zadáním příkazu `format 600MB` se vytvoří soubor, jehož název je zadán v parametru, a připraví ho k použití. V této fázi je již možné používat všechny příkazy implementované nad souborovým systémem (viz kapitola 1).

Při dalším spuštění již bude soubor existovat a bude obsahovat námi vytvořené soubory a adresáře a dojde pouze k načtení obsahu do programu.

2.2.3 Příkazy programu

Jednotlivé příkazy nad souborovým systémem jsou podrobně uvedeny v kapitole 1.

Stručný přehled příkazů:

- `cp s1 s2` – zkopíruje soubor `s1` do umístění `s2`,
- `mv s1 s2` – přesune soubor `s1` do umístění `s2`,
- `rm s1` – smaže soubor `s1`,
- `mkdir a1` – vytvoří adresář `a1`,
- `rmdir a1` – smaže prázdný adresář `a1`,
- `ls a1` – vypíše obsah adresáře `a1`,
- `cat s1` – vypíše obsah souboru `s1`,
- `cd a1` – změní aktuální cestu do adresáře `a1`,

- `pwd` – vypíše aktuální cestu,
- `info a1/s1` – vypíše informace o adresáři/souboru,
- `incp s1 s2` – nahraje soubor `s1` z pevného disku do umístění `s2` v pseudoNTFS,
- `outcp s1 s2` – nahraje soubor `s1` z pseudoNTFS do umístění `s2` na pevném disku,
- `load s1` – sekvenčně vykoná příkazy umístěné v souboru `s1` na pevném disku,
- `format <velikost> MB` – zformátuje soubor na zadanou velikost,
- `slink s1 s2` – vytvoří symbolický link na soubor `s1` s názvem `s2`,
- `exit` – uvolní paměť, zavře proudy a ukončí program.

Příkaz `slink`

Příkaz `slink s1 s2` vytvoří symbolický link na soubor `s1` s názvem `s2`. Symbolický link lze vytvořit pouze mezi obyčejnými soubory.

Nad symbolickým linkem lze vykonávat následující příkazy:

- `cp s1 s2` – vytvoří kopii symbolického linku `s1` do umístění `s2` (bude odkazovat na stejný soubor),
- `mv s1 s2` – přesune symbolický link `s1` do umístění `s2`,
- `rm s1` – smaže symbolický link `s1`,
- `cat s1` – vypíše obsah souboru, na který odkazuje symbolický link `s1`,
- `info s1` – vypíše informace o symbolickém linku `s1`,
- `outcp s1 s2` – nahraje soubor, na který odkazuje symbolický link `s1`, do umístění `s2` na pevném disku.

Symbolický link je implementován tak, že v MFT záznamu je vytvořen speciální typ položky: *symbolický link*. Tento typ položky obsahuje pouze jeden fragment délky 1, protože v datové části je uloženo pouze UID souboru (`integer` – 4 byty), na který symbolický link odkazuje, a ukončovací hodnota `-1` (`integer` – 4 byty). Díky tomu je velikost symbolického linku vždy 8 bytů a v datové části pro něj postačuje jen jeden cluster.

2.2.4 Překlad programu

Pro jednoduchý překlad je v kořenovém adresáři programu vytvořen soubor `makefile`. Překlad je spuštěn zadáním příkazu `make` do *Příkazového řádku*. Překlad je realizován překladačem `gcc`.

K překladu je nutné mít nainstalovány programy: `make`, `gcc`.

3 Uživatelská dokumentace

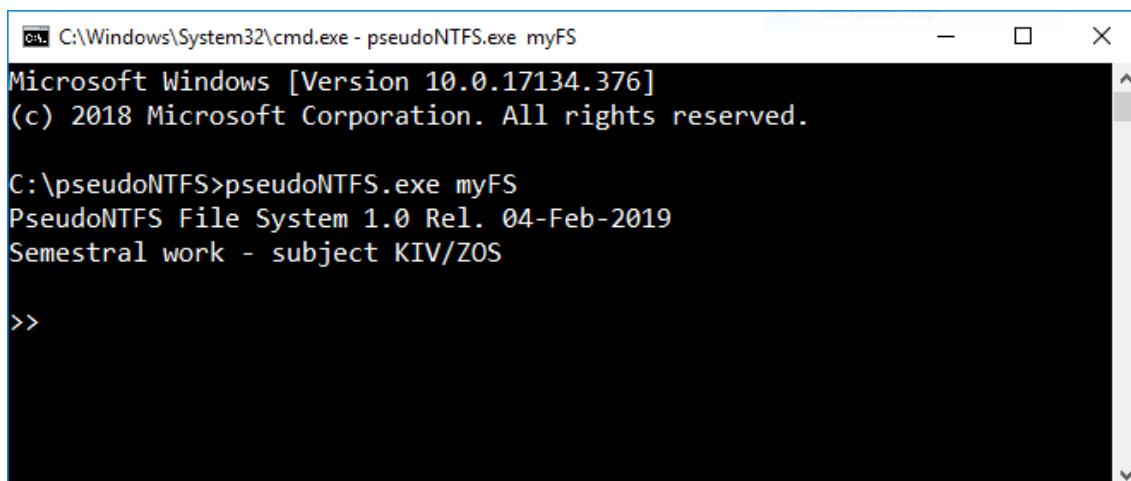
V následující kapitole je popsáno spuštění a ovládání souborového systému *pseudoNTFS*.

3.1 Spuštění programu

Jedna z možných variant spuštění programu je popsána v následujících bodech:

1. otevřít *Příkazový řádek* a dostat se do kořenového adresáře programu,
2. spustit program příkazem: `pseudoNTFS.exe <souborový_systém>`.

Parametr *<souborový_systém>* je povinný. Jedná se o název souboru, do kterého bude ukládán souborový systém. Pokud bude program spuštěn bez parametru, program vypíše náповědu o správném použití. Ilustrační příklad spuštění programu je zobrazen na obrázku 3.1.



```
C:\Windows\System32\cmd.exe - pseudoNTFS.exe myFS
Microsoft Windows [Version 10.0.17134.376]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\pseudoNTFS>pseudoNTFS.exe myFS
PseudoNTFS File System 1.0 Rel. 04-Feb-2019
Semestral work - subject KIV/ZOS

>>
```

Obrázek 3.1: Příklad spuštění programu

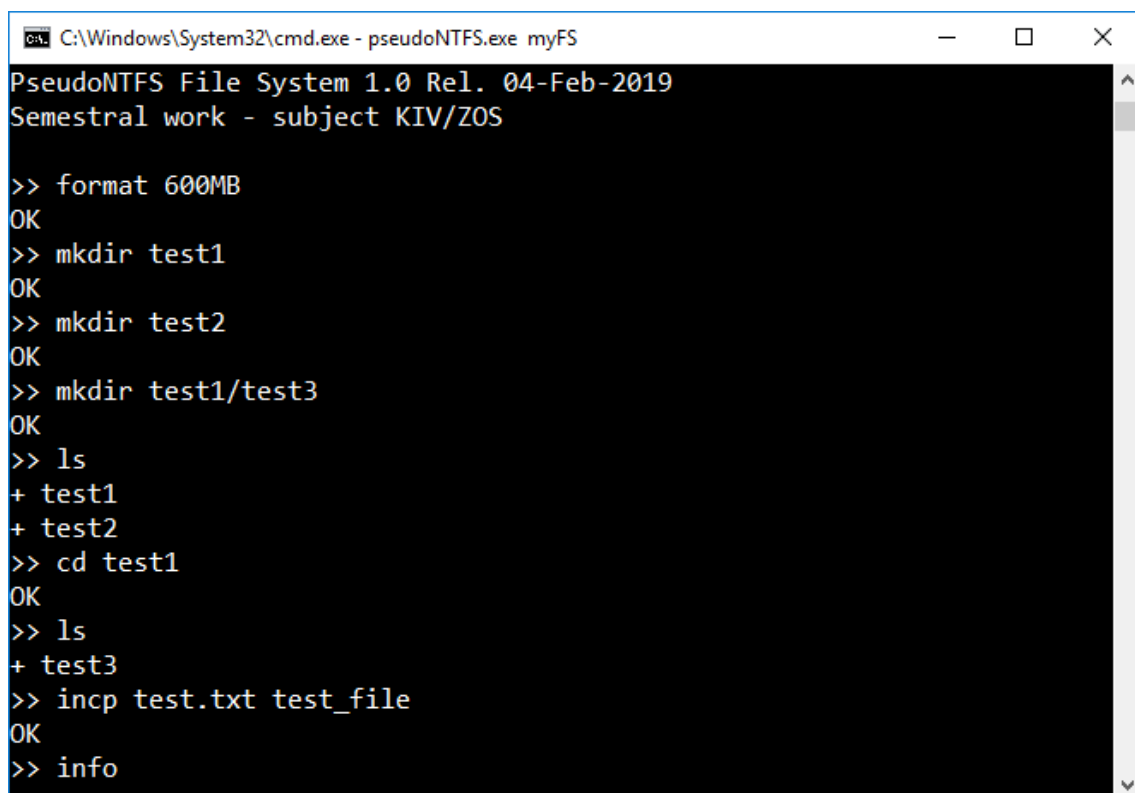
3.2 Ovládání programu

Po úspěšném spuštění programu mohou nastat dvě možnosti: buď soubor předaný v parametru existuje, nebo neexistuje.

V případě, že soubor neexistuje, je nutné jej nejprve vytvořit a zformátovat na počáteční hodnoty. O to se stará příkaz `format <velikost> MB`. Po zformátování souboru je možné používat všechny příkazy uvedené v kapitole 1.

Pokud již soubor, jehož název je předaný v parametru při spuštění programu, existuje, dojde k načtení obsahu tohoto souboru a je možné používat příkazy uvedené v kapitole 1 ihned.

Program je možné ukončit zadáním příkazu `exit`. Ilustrační příklad běhu programu je zobrazen na obrázku 3.2.



```
C:\Windows\System32\cmd.exe - pseudoNTFS.exe myFS
PseudoNTFS File System 1.0 Rel. 04-Feb-2019
Semestral work - subject KIV/ZOS

>> format 600MB
OK
>> mkdir test1
OK
>> mkdir test2
OK
>> mkdir test1/test3
OK
>> ls
+ test1
+ test2
>> cd test1
OK
>> ls
+ test3
>> incp test.txt test_file
OK
>> info
```

Obrázek 3.2: Příklad běhu programu

4 Závěr

4.1 Testování

Program byl testován na operačních systémech:

- Windows 10 Enterprise,
- Ubuntu 18.04.1 LTS (Bionic Beave),
- Debian GNU/Linux 9 (stretch).

V rámci testování došlo například k porovnání, zda při importu a následném exportu daného souboru jsou oba soubory stejně velké. Dále jsem testoval schopnost programu vypořádat se s nevalidními vstupy od uživatele, nahrání velkých souborů do souborového systému a také nahrání velkého množství různě velkých souborů. Program ve zmíněných testech uspěl.

4.2 Zhodnocení práce

Cílem semestrální práce bylo navrhnout a implementovat souborový systém pseudoNTFS, který bude umět zpracovávat základní příkazy nad soubory a adresáři (viz kapitola 1).

Návrh a implementace řešení proběhly úspěšně. Při řešení této semestrální práce jsem se nesetkal s většími problémy.

Semestrální práce z mého pohledu splňuje požadavky zadání.