

Semestrální práce ZOS 2017

Práce se souborovým systémem pseudoNTFS

30.01.2018

Luboš Hubáček, P15B0072P

Obsah

[1. Zadání práce 1](#_Toc505099426)

[1.1 Seznam validních příkazů 1](#_Toc505099427)

[2. Parametry pseudo NTFS 2](#_Toc505099428)

[3. Struktura aplikace 2](#_Toc505099429)

[3.1 Základní činnost aplikace 2](#_Toc505099430)

[3.2 Jak aplikace pracuje s daty 2](#_Toc505099431)

[4. Spuštění programu 3](#_Toc505099432)

[4.1 Stažení programu 3](#_Toc505099433)

[4.2 Překlad programu 3](#_Toc505099434)

[4.2 Spuštění programu 3](#_Toc505099435)

[5. Závěr 4](#_Toc505099436)

# Zadání práce

Tématem semestrální práce je práce se souborovým systémem pseudoNTFS. Cílem je splnit několik vybraných úloh, které jsou podobněji specifikovány v dokumentech, na které je odkazováno níže.

Implementace probíhá v programovacím jazyce C.

**Aktuální zadání práce lze stáhnout na URL:** <https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=139572>

**Poznámky k zadání lze stáhnout na URL:**

<https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=140060>

## Seznam validních příkazů

* cp s1 s2 Kopírování souborů
* mv s1 s2 Přesun souborů
* rm s1 Smazání souboru
* mkdir a1 Vytvoří složku
* rmdir a1 Smaže prázdný adresář
* ls a1 Vypíše obsah adresáře
* cat s1 Vypíše obsah souboru
* cd a1 Změní aktuální cestu do adresáře
* pwd Vypíše aktuální cestu
* info a1/s1 Vypíše informace o souboru
* incp s1 s2 Nahraje soubor z pevného disku
* outcp s1 s2 Nahraje soubor z pseudoNTFS
* load s1 Vykoná sekvenci příkazů ze souboru
* consist Kontrola konzistence
* defrag Defragmentace
* exit Ukončení programu

# Parametry pseudo NTFS

Jako parametry mé implementace pseudo souborového systému jsem zvolil:

* Počet clusterů 200
* Velikost clusteru 100
* Počet fragmentů pro jeden mft item 10
* Název domovské složky ROOT\_DIR
* ID domovské složky 0

# Struktura aplikace

Pro implementaci jsem zvolil programovací jazyk C, kde jsem jednotlivé funkcionality dekomponoval do jednotlivých modulů. Tím jsem docílil větší přehlednosti zdrojového kódu.

Rozložení demonstruje následující tabulka:

|  |  |
| --- | --- |
| Název souboru | Popisek |
| boot\_record | Struktura pro uchování boot\_recordu |
| debugger | Obsahuje kód speciálního nástroje pro debugging projektu |
| functions | Užitečné funkce, které pracují na vyšší úrovni (komplexnější) |
| loader | Načtení nebo vytvoření základního souboru se souborovým systémem |
| main | Hlavní třída aplikace, slouží ke spouštění |
| mft | Uchovávání MFTI a MFTF |
| ntfs\_helpers | Nízkoúrovňové funkce pro práci s daty uloženými ve FS |
| parametr | Sdílená paměť využívaná při kontrole konzistence |
| shell | Zpracování jednotlivých příkazů z klávesnice |
| shell\_functions | Obsahuje handlery pro zadávané příkazy |

## 3.1 Základní činnost aplikace

Po spuštění programu *loader* ověří, jestli existuje souborový systém v požadovaném umístění, v případě, že neexistuje, tak dojde k jeho vytvoření s výchozími parametry, které jsou popsány v kapitole 2. Následně dojde k otevření souborového systému, jehož základní data se nahrají do paměti (*boot record, MFT tabulka, bitmapa*).

Následně je spuštěn *loader*, který čeká na příkazy zadávané s klávesnice. Po zadání příkazu se ověří existence příkazu a dojde případně ke spuštění příslušné části zdrojového kódu. V případě, že je zadán validní příkaz, tak *shell* spustí obslužný podprogram, který se nachází v modulu *shell\_functions*. Tyto obslužné podprogramy si pak následně pouští funkce z ostatních knihoven dle svých potřeb.

## 3.2 Jak aplikace pracuje s daty

Všechna data kromě obsahu clusterů si držím celou dobu aktuální v paměti a při manipulaci s daty nejprve provedu manipulaci na načtených datech a hned tyto změny zapisuji i do souboru. Tímto mám zajištěno, že při případném nekorektním ukončení práce s programem nedojde ke ztrátě dat.

# Spuštění programu

## 4.1 Stažení programu

Aplikaci je možno stáhnout z repozitáře na GitHubu: <https://github.com/Sekiphp/KIV-ZOS>

## 4.2 Překlad programu

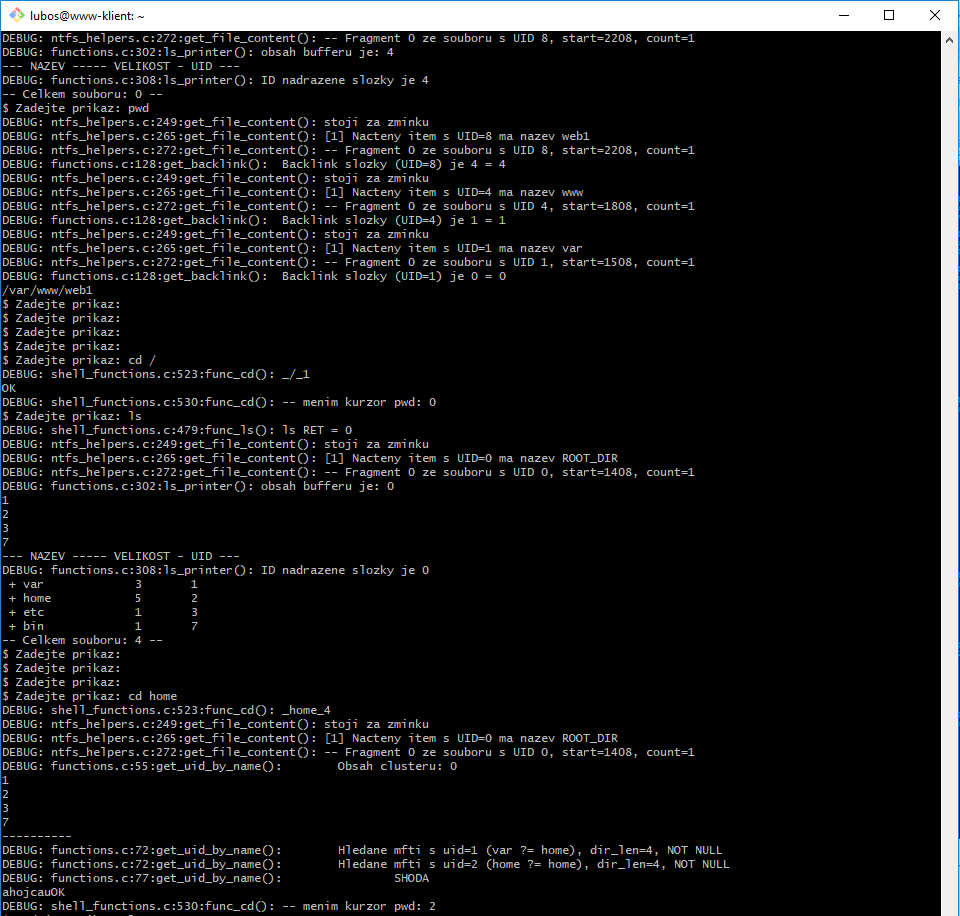
K přeložení programu dojde po zadání příkazu: **gcc \*.c -o pseudontfs -pthread -Wall**

Alternativním způsobem pro přeložení programu je využití utility *makefile* prostým zadáním příkazu: **make**

*(Poznámka: při použití příkazu make dojde bezprostředně po překladu ke spuštění aplikace)*

## 4.2 Spuštění programu

Pro spuštění programu je možné využít utility *make*, nebo zadat příkaz: **./pseudontfs ntfs.dat**

****

Obrázek : Ukázka spuštění programu v debug režimu

# Závěr

Podařilo se mi implementovat poměrně komplexní celek, který reprezentuje virtuální souborový systém NTFS. Výsledný program jsem testoval a vyvíjel na Linuxu, jelikož nativní podporu pro OS Windows jsem nezamýšlel. Nejspíš by ale netrvalo mnoho času a tato podpora by se dala poměrně snadno doimplementovat.

Realizace práce byla o dost delší, než jsem si původně myslel. Veškerý svůj čas strávený s implementací práce jsem si pečlivě měřil a dostal jsem se na **84 hodin**.

Aplikace splňuje všechny prvky kladené zadáním.