FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Semestrální práce ZOS 2025

Zjednodušený souborový systém založený na i-uzlech

Autor: Daniel Rössner
Program: `fs_app`
Soubor VFS: `.vfs`
Datum: 4. února 2026

Obsah

Kapitola 1

Zadání

Tématem práce je implementace zjednodušeného souborového systému založeného na i-uzlech, uloženého v jednom souboru na hostitelském systému. Program se spouští s jedním parametrem (cesta k souboru VFS) a následně v interaktivním režimu přijímá příkazy.

Základní funkčnost dle zadání:

- `cp s1 s2` – kopie souboru v rámci VFS
- `mv s1 s2` – přesun/přejmenování v rámci VFS
- `rm s1` – smazání souboru
- `mkdir a1` – vytvoření adresáře
- `rmdir a1` – smazání prázdného adresáře
- `ls [a1]` – výpis obsahu adresáře (výchozí aktuální)
- `cat s1` – výpis obsahu textového souboru
- `cd a1` – změna aktuálního adresáře
- `pwd` – výpis aktuální cesty
- `info s1/a1` – informace o položce a použitých clusterech
- `incp host_path vfs_path` – import souboru z OS do VFS
- `outcp vfs_path host_path` – export souboru z VFS do OS
- `load host_file` – načtení příkazů ze souboru (1 příkaz / řádek)
- `format SIZE` – vytvoření/naformátování VFS na danou velikost (např. `600MB`)
- `statfs` – statistiky souborového systému
- `exit` – ukončení programu

Volitelná část pro login začínající písmeny `j-r` (v této implementaci využita):

- `xcp s1 s2 s3` – vytvoří `s3` jako spojení `s1` a `s2`
- `add s1 s2` – přidá na konec `s1` obsah `s2`

Formát výstupů je závazný (např. `OK`, `FILE NOT FOUND`, `PATH NOT FOUND`, `NOT EMPTY`, `EXIST`) a je zachován tak, aby program šel automaticky testovat.

Kapitola 2

Analýza úlohy

2.1 Klíčové požadavky a omezení

Zadání explicitně stanovuje:

- maximální délku názvu položky: **12 bytů** (formát 8+3+\0); kratší názvy se doplňují nulami,
- práci se souborovým systémem uloženým v **jednom souboru** (VFS),
- podporu absolutních i relativních cest,
- existenci adresářů a souborů (adresáře obsahují položky typu `directory_item`),
- clusterovou organizaci dat a bitmapy obsazení.

Implementace používá pevnou velikost clusteru **1024 B**, což zjednodušuje výpočty a adresaci. Souborový systém je minimalistický: každá položka je reprezentována *pseudo-iuzlem* (struktura `pseudo_inode`), data jsou uložena v datové oblasti a volné/obsazené clustery a i-uzly jsou spravovány bitmapami.

2.2 Použité datové struktury

2.2.1 Superblock

Superblock je uložen na začátku VFS a obsahuje základní metadata (velikost disku, velikost clusteru, počty clusterů) a také *adresy začátků* jednotlivých oblastí v souboru: bitmapa i-uzlů, bitmapa datových bloků, tabulka i-uzlů a datová oblast.

2.2.2 Pseudo-iuzel

Pseudo-iuzel (`struct pseudo_inode`) obsahuje:

- identifikátor i-uzlu (`nodeid`),
- příznak adresáře (`isDirectory`),
- počet referencí (`references`) – využitelné např. pro hardlink (v této práci je využito jako počítadlo),
- velikost souboru v bytech (`file_size`),

- pět přímých odkazů na datové clustery (`direct1..direct5`),
- dva nepřímé odkazy (`indirect1, indirect2`) – rezervováno pro budoucí rozšíření.

V aktuální implementaci je pro data souborů využíváno **max. 5 přímých clusterů** (tj. typicky do 5 KiB). Pole `indirect1/2` je udržováno, ale není aktivně používáno pro zvětšení maximální velikosti souboru.

2.2.3 Položka adresáře

Adresář je uložen jako soubor, jehož data jsou tvořena záznamy `struct directory_item`:

- `inode` – ID cílového i-uzlu,
- `item_name[12]` – název položky (12 bytů).

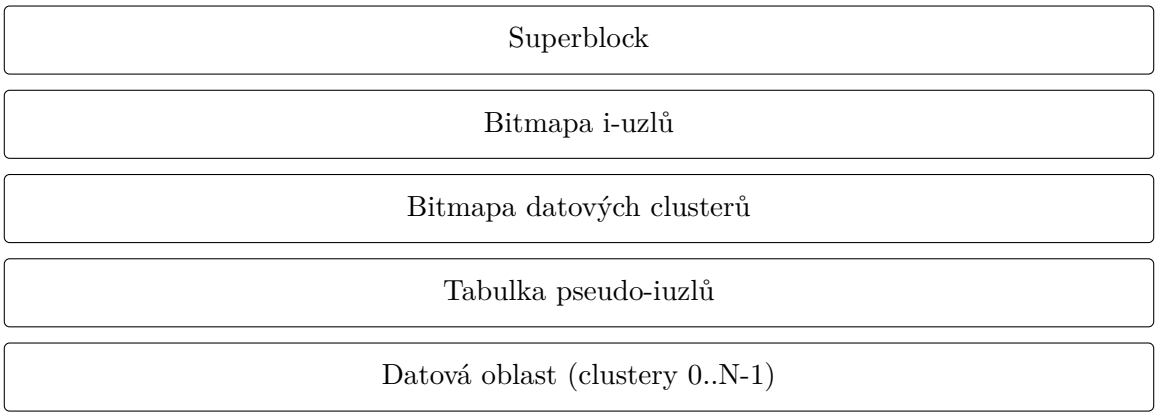
Každý adresář obsahuje minimálně položky `.` a `..`. Kořenový adresář je vytvořen při formátování.

Kapitola 3

Návrh řešení

3.1 Rozložení dat ve VFS

Soubor `.vfs` je rozdělen na čtyři hlavní oblasti:



Adresy začátků oblastí jsou uloženy v superblocku. Při `format` se velikosti bitmap dopočítají z počtu clusterů (zaokrouhlení na bity).

3.2 Inicializace souborového systému

Příkaz `format`:

- vytvoří (nebo přepíše) soubor VFS na požadovanou velikost,
- zapíše superblock a obě bitmapy,
- inicializuje tabulku i-uzlů (`inode 0` je kořenový adresář),
- alokuje datový cluster 0 pro kořenový adresář a zapíše do něj položky `.` a `..`

3.3 Adresace a cesty

Program podporuje:

- **absolutní cesty** (začínají `/`) – vyhodnocují se od kořene,

- **relativní cesty** – vyhodnocují se od aktuálního pracovního adresáře.

Překlad cesty na ID i-uzlu je realizován postupným procházením adresářových položek (`find_inode_in_dir`) a parsováním komponent (oddělovač `/`).

Kapitola 4

Implementace

4.1 Struktura projektu

Projekt je rozdělen do dvou hlavních složek:

- `include/` – hlavičkové soubory (`structs.h`, `fs_core.h`, `fs_utils.h`)
- `src/` – implementace jednotlivých příkazů a pomocných funkcí

4.2 Moduly a odpovědnosti

4.2.1 `src/main.c`

- Parsuje argumenty programu (cesta k VFS).
- Implementuje interaktivní smyčku (čtení řádku, tokenizace na příkaz a parametry).
- Udržuje aktuální pracovní adresář (pro `cd`, `pwd`).
- Směruje příkazy do příslušných funkcí (`fs_mkdir`, `fs_rm`, `fs_incp`, ...).

4.2.2 `src/cmd_system.c`

Obsahuje systémové příkazy:

- `fs_format` – vytvoření FS, výpočet layoutu a inicializace rootu,
- `fs_statfs` – výpočet a výpis statistik (velikost, volné/obsazené bloky a i-uzly, počet adresářů),
- `fs_info` a `fs_info_path` – výpis informací o i-uzlu včetně použitých přímých odkazů.

4.2.3 `src/cmd_dir.c`

Adresářové operace:

- `fs_mkdir` – založení adresáře (alokace i-uzlu a 1 datového clusteru, zápis `.` a `..`),
- `fs_rmdir` – kontrola prázdnosti a následné uvolnění,
- `fs_ls` – výpis položek (formát `FILE: ...` / `DIR: ...`),
- `fs_path_to_inode` – překlad cesty (využívá utilitní funkce).

4.2.4 `src/cmd_file.c`

Souborové operace:

- `fs_cat` – načtení obsahu souboru a výpis,
- `fs_rm` – odstranění souboru a uvolnění zdrojů,
- `fs_cp` – kopie souboru uvnitř VFS (na úrovni obsahu),
- `fs_mv` – přesun/přejmenování (úprava položek v adresáři),
- `fs_incp`/`fs_outcp` – import/export mezi host OS a VFS.

4.2.5 `src/cmd_extra.c`

Rozšířené příkazy pro login `j-r`:

- `fs_xcp` – spojení dvou souborů do třetího,
- `fs_add` – připojení obsahu souboru na konec jiného.

4.2.6 `src/fs_utils.c`

Sada nízkoúrovňových pomocných funkcí:

- práce se superblockem a i-uzly (`load_superblock`, `read_inode`, `write_inode`),
- práce s bitmapami (`find_free_bit`, `set_bit`),
- práce s adresáři (`add_directory_item`, `remove_directory_item`, `is_dir_empty`),
- parsování cest a základní „bufferové“ operace pro copy/concat.

4.3 Důležité implementační detaily

4.3.1 Alokace i-uzlu a datového clusteru

Alokace probíhá vždy přes bitmapu:

1. nalezení volného bitu (`find_free_bit`),
2. nastavení bitu na obsazeno (`set_bit`),
3. zápis odpovídající struktury (i-uzel, data).

4.3.2 Mazání a uvolnění zdrojů

Mazání souboru/adresáře uvolňuje:

- položku v rodičovském adresáři (jméno → inode),
- datové clustery z přímých odkazů,
- samotný inode v bitmapě i-uzlů.

Pro `rmdir` je navíc vyžadováno, aby byl adresář prázdný (obsahoval pouze `.` a `..`).

4.3.3 Chování při chybách

Program předpokládá syntakticky korektní příkazy, ale kontroluje sémantiku:

- neexistující cesta/položka → `PATH NOT FOUND` / `FILE NOT FOUND`,
- pokus o založení existujícího adresáře → `EXIST`,
- pokus o smazání neprázdného adresáře → `NOT EMPTY`.

Kapitola 5

Uživatelská příručka

5.1 Překlad a spuštění

K dispozici je `Makefile`. Typický postup:

```
make
./fs_app disk.vfs
```

5.2 Interaktivní režim

Po spuštění program čeká na příkazy. Příklady:

```
./fs_app disk.vfs
format 10MB
OK
mkdir in
OK
incp README.txt /in/readme.txt
OK
ls /in
FILE: readme.txt
```

5.3 Popis příkazů

Příkaz	Popis
`format SIZE`	Naformátuje VFS na velikost `SIZE` (např. `600MB`).
`statfs`	Vypíše souhrn statistik FS (velikost, bloky, i-uzly, adresáře).
`pwd`	Vypíše aktuální cestu.
`cd PATH`	Změní aktuální adresář.
`ls [PATH]`	Vypíše obsah adresáře (výchozí aktuální).
`mkdir PATH`	Vytvoří nový adresář.
`rmdir PATH`	Smaže prázdný adresář.
`info PATH`	Vypíše informace o položce a použitých clusterech.
`cat PATH`	Vypíše obsah souboru na STDOUT.
`incp H V`	Import: host soubor `H` → VFS cesta `V`.
`outcp V H`	Export: VFS soubor `V` → host cesta `H`.

`cp S1 S2`	Kopie souboru v rámci VFS.
`mv S1 S2`	Přesun/přejmenování v rámci VFS.
`rm PATH`	Smazání souboru.
`xcp S1 S2 S3`	Spojí `S1` a `S2` do `S3`.
`add S1 S2`	Připojí obsah `S2` na konec `S1`.
`load FILE`	Načte příkazy ze souboru a provede je.
`exit`	Ukončí program.

Kapitola 6

Testování

6.1 Ruční testy

Testy byly prováděny v interaktivním režimu nad naformátovaným VFS:

- založení adresářové struktury (`mkdir`, `cd`, `pwd`, `ls`),
- import a export souborů (`incp`, `outcp`),
- práce se soubory (`cp`, `mv`, `rm`, `cat`),
- kontrola hraničních stavů (`EXIST`, `NOT EMPTY`, neexistující cesty),
- rozšířené příkazy (`xcp`, `add`),
- ověření konzistence (`statfs`, `info`).

6.2 Testovací scénáře

```
format 1MB
mkdir in
incp host.txt /in/a.txt
cp /in/a.txt /in/b.txt
cat /in/b.txt
mv /in/b.txt /in/c.txt
rm /in/c.txt
rmdir /in # musi selhat pokud neni prazdny
```

Kapitola 7

Závěr

Implementace splňuje požadované příkazy pro zjednodušený souborový systém s i-uzly a podporuje práci s cestami, adresáři a základními operacemi se soubory. Navíc obsahuje rozšířené příkazy `xcp` a `add`. Struktura kódu je rozdělena do logických modulů a doplněna dokumentačními komentáři.

Známá omezení

- Maximální velikost souboru je v této verzi omezena pěti přímými clustery (typicky do 5 KiB).
- Nepřímé odkazy (`indirect1/2`) jsou v datových strukturách připraveny, ale nejsou využity pro alokaci dalších bloků.