FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV/ZOS

VIRTUÁLNÍ SOUBOROVý SYSTÉM ZALOžENý NA I-UZLECH

Jakub Vítek (viteja) - A19B0222P

Leden 2020

Obsah

Semestrální práce ZOS 2019 (verze dokumentu 01)

Tématem semestrální práce bude práce se zjednodušeným souborovým systémem založeným na i-uzlech. Vaším cílem bude splnit několik vybraných úloh.

Základní funkčnost, kterou musí program splňovat. Formát výpisů je závazný.

Program bude mít jeden parametr a tím bude název Vašeho souborového systému. Po spuštění bude program čekat na zadání jednotlivých příkazů s minimální funkčností viz níže (všechny soubory mohou být zadány jak absolutní, tak relativní cestou):

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 clusterech se nachází)

```
info a1/s1
Možný výsledek:
```

```
NAME – SIZE – i-node NUMBER – přímé a nepřímé odkazy
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 spuštení 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
```

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

Informace k zadání a omezením

- Maximální délka názvu souboru bude 8+3=11 znaků (jméno.přípona) + \0 (ukončovací znak v C/C++), tedy 12 bytů.
- Každý název bude zabírat právě 12 bytů (do délky 12 bytů doplníte \0 - při kratších názvech).

Nad vytvořeným a naplněným souborovým systémem umožněte provedení následujících operací:

- Defragmentace (defrag) – pokud login studenta začíná **a-i**
  Datové bloky budou organizovány tak, že nejprve budou obsazené a následně volné (předpokládáme dostatek volného místa – minimálně ve velikosti největšího souboru).

- Kontrola konzistence (check) – pokud login studenta začíná **j-r**
  Zkontrolujte, zda jsou soubory nepoškozené (např. velikost souboru odpovídá počtu alokovaných datových bloků) a zda je každý soubor v nějakém adresáři. Součástí řešení bude nasimulovat chybový stav, který následná kontrola odhalí.

- Symbolický link (slink s1 s2) – pokud login studenta začíná **s-z**
  Vytvoří symbolický link na soubor s1 s názvem s2. Dále se s ním pracuje očekávaným způsobem, tedy např. cat s2 vypíše obsah souboru s1.

Odevzdání práce

Práci včetně dokumentace pošlete svému cvičícímu e-mailem. V případě velkého objemu dat můžete využít různé služby (leteteckaposta.cz, uschovna.cz).

Osobní předvedení práce cvičícímu. Referenčním strojem je školní PC v UC326. Práci můžete ukázat i na svém notebooku. Konkrétní datum a čas předvedení práce si domluvte e-mailem se cvičícím, sdělí vám časová okna, kdy můžete práci ukázat.

Do kdy musím semestrální práci odevzdat?

- Zápočet musíte získat do mezního data pro získání zápočtu (10. února 2020).
- A samozřejmě je třeba mít zápočet dříve, než půjdete na zkoušku (alespoň 1 den předem).

Hodnocení

Při kontrole semestrální práce bude hodnocena:

- Kvalita a čitelnost kódu včetně komentářů
- Funkčnost a kvalita řešení
- Dokumentace

1 Aplikace

Aplikace byla primárně vyvíjena, ve standardu C99 programovacího jazyka C, pro operační systém GNU/Linux (Fedora 31), měla by však podporovat všechny operační systémy podporující standard POSIX. Pro kompatibilitu s operačním systémem Windows byla část zdrojového kódu reimplementována či nahrazena, kompatibilita je docílena využitím podmínek a maker preprocessoru jazyka C.

Aplikace má několik základních omezení v parametrech. Některá omezení jsou umělá, například dodržení maximální velikosti souboru 12 byte, spousta omezení však vyplývá z implementace aplikace za pomocí 32 bitových hodnot. Zbylé hodnoty by mohly být měnitelné (například velikost datového bloku), byly však pěvně implementované.

Program je spustitelným binárním souborem, který má jeden povinný vstupní parametr - cestu k virtuálnímu souborovému systému (VFS). V případě, že existuje rodičovská složka a neexistuje soubor VFS, soubor bude vytvořen a naformátován na velikost 64kB.

2 Struktura

Struktura souboru je závislá na požadované velikosti.

2.1 Superblok

Virtuální souborový systém obsahuje jeden superblok, který je umístěn v hlavičce na začátku souboru VFS a jeho velikost je 284 byte. Tato struktura obsahuje základní informace o umístění jednotlivých částí VFS. Defici struktury lze vidět v hlavičkovém souboru *superblock.h*.

2.2 Hlavička a datová část

Velikost hlavičky souborového systému se odvíjí od celkové velikosti souboru. Na hlavičku jsou pevně vyhrazená 4% celkové velikosti systému (např. 600MB soubor => 600MB * 4% = 24MB). Hlavička obsahuje (v tomto pořadí): *superblok*, *bitmapu*, *prostor pro i-uzly*. Formátování souborového systému proběhne úspěšně i v případě, že prostor pro hlavičku je příliš malý, například když se nezapíše dostatečné množství byte pro bitmapu či pro i-uzly. Takto malé systémy jsou však nepoužitelné. Zbylých 96% je využito pro ukládání dat.

2.3 Bitmapa

Bitmapa je součástí hlavičky souborového systému a její velikost je na celkové velikosti VFS závislá. Jeden blok bitmapy je velký 1 byte a může obsahovat hodnotu 0 (indikace volného databloku), hodnotu 1 (indikace využitého databloku) či jinou chybovou hodnotu. Počet bloků bitmapy je při dostatku místa pro VFS celkovým počtem datových bloků.

2.4 I-uzly

Po té, co do vyhrazených 4% hlavičky VFS jsou zapsány superblok a bitmapa, je zbylé volné místo využito na uložení i-uzlu. Samotný i-uzel má 44 Byte, některé ukazatele jsou však uloženy do datových bloků jako první či druhý nepřímý ukazatel. Na obsah virtuálního souborového systému je od počáteční adresy pro i-uzly do počátku datové části nahlíženo jako na pole. I-uzly jsou číslovány dle jejich pořadí zápisu.

2.5 Datová část

Zbylá část virtuálního souborového systému obsahuje místo, pro uložení dat. Toto místo je rozděleno na datové bloky. Jeden datový blok má v současné implementaci velikost 4096 byte. Indikace, zda je datový blok využíván, je umístěna v bitové mapě. Nultý datový blok je nultým blokem bitmapy.

3 Input/Output

Virtuální souborový systém v rámci této aplikace je standardním souborem na hostitelském operačním systému (GNU/Linux, Windows), který dodržuje strukturu popsanou v předchozí kapitole. Aplikace vnitřně pro všechny úpravy používá knihovnu *stdio.h* (funkce fopen, fseek, fwrite, fread, fclose). Vzhledem k tomu, že nahrávané soubory nemusí být umístěny kontiunálně na sobě navazujících datových blocích, bylo třeba přístup k datům standardizovat, jinak by zdrojový kód pro práci s vnitřními VFS soubory byl extrémně opakován a zvyšoval nečitelnost.

Z tohoto důvodu byla provedena zjednodušená reimplementace výše uvedených funkcí - *VFS_IO.h* (VFS_FILE, VFS_open, VFS_seek, VFS_read, VFS_write, VFS_close). Tato miniknihovna umožnuje pracovat s vnitřním souborem VFS dostatečně podobným způsobem jako v případě stdio.h.

4 Soubory

V tomto virtuálním souborovém systému platí, že jeden i-uzel se rovná jednomu souboru bez ohledu na typ souboru. Každý soubor má přiděleny přímé či nepřímé datové bloky, které určují jeho prostor pro čtení a zápis dat. Soubory se virtuálně

Souborový systém má tři základní typy souborů.

4.1 Soubor

Obyčejný binární soubor bez speciálně definované struktury.

4.2 Složka

Složka je souborem. Každých 16 byte tohoto souboru lze převést na data struktury *directory_entry*. Každá složka má minimálně dvě tyto struktury, což znamená, že minimální velikost složky je 32 byte. První dvě struktury obsahují: záznam pro aktuální složku a záznam rodičovské složky.

Složky virtuálně vytváří stromovou strukturu, která začíná kořenovou složkou, která má ID = 1 a první dva záznamy ve složce stejné.

4.3 Symbolický link

Symbolický link je soubor obsahující řetězec, který je cestou k souboru, na který symbolický link ukazuje.

5 Příkazový řádek

Příkazový řádek dovoluje interakci uživatele s virtuálním souborovým systémem. Příkazový řádek je cyklem, který vypíše cestu složku, do které je uživatel právě přepnut (před znakem >). Následně je očekáván vstup od uživatele (jeden z příkazů uvedených v zadání). Vstup je zpracován a probíhá ověření, jestli se nejedná o rozpoznaný příkaz. V případě, že je příkaz rozpoznán, zpracovaný vstup od uživatele je předán funkci, která ověří požadované parametry a provede akci.

Seznam příkazů lze vidět v zadání práce. K těmto příkazům byl navíc doplněn příkaz exit, jež bezpečně ukončí program (ukončení lze také provést zkratkou ctrl+c [SIGINT] ve chvíli kdy příkazový řádek čeká na vstup)

6 Uživatelská příručka

6.1 Překlad a sestavení

Aplikaci lze přeložit, sestavit a spustit pod operačními systémy GNU/Linux (instalujte: gcc, make dle pokynů Vaší distribuce) a Windows (instalujte: MinGW - gcc, make).
Aplikaci je doporučeno sestavit přiloženými Makefile (Makefile.win pro windows).

6.2 Spuštění

Po překladu a sestavení pomocí přiložených makefile lze aplikaci spustit příkazem `./KIV_ZOS <cesta_k_vfs_souboru>`. Parametr aplikace je povinný. Cesta k rodičovské složce VFS souboru musí existovat, soubor samotný nikoliv - v případě, že soubor neexistuje, bude vytvořen a naformátován na velikost 64KB.

6.3 Ovládání

Aplikace žádá od uživatele vstupní příkazy v podobném duchu jako při práci se soubory v operačním systému GNU/Linux. Seznam příkazů je možné přečíst v zadání práce.

7 Závěr

Implementoval jsem relativně komplexní aplikaci (na poměry toho, co se obvykle dělá v předchozích předmětech na FAV ZČU). Aplikaci jsem primárně vyvíjel a testoval na operačním systému GNU/Linux s podporou a testováním na systému Microsoft Windows.

Aplikace má mnoho nedostatků, při vývoji jsem bohužel zvolil 32 bitová čísla, jež mě značně omezují. Dále jsem při vývoji zahodil možnost rozsáhlejší nastavitelnosti programu (např. velikost datového bloku). Aplikace striktně běží na jediném vláknu, čtení a zápis nemají svou vyrovnávací pamět, takže se rychlost aplikace zpomaluje čekáním na I/O operace.

I přes tyto nedostatky však aplikace splňuje všechny požadavky.