FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z předmětu

Základy operačních systémů

Pseudo Filesystem založený na i-uzlech

10. prosince 2023

Autor:
Václav Tran
A21B0299P
nuva@students.zcu.cz

Obsah

Kapitola 1

Zadání

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 (všechny soubory mohou být zadány jak absolutní, tak relativní cestou).

Podrobnější popis viz [1].

Kapitola 2

Implementace

Semestrální práce byla implementovaná v programovacím jazyce Go. V celé aplikaci se hojně využívá konstrukce `slice`[1].

2.1 Popis aplikace

Aplikace se skládá ze souborů:

- `command_parser.go` - provádí transformaci vstupu uživatele na příkazy
- `command_interpreter.go` - intepretuje příkazy a zpracovává je
- `fs_commands.go` - obsahuje veškeré metody pro práci (tj. vytváření i-uzlů, ukládání dat apod.) s filesystémem
- `fs_structs.go` - potřebné struktury pro správné fungování aplikace
- `main.go` - obsahuje startovací bod aplikace

2.1.1 Metody `command_parser.go`

Tato sekce popisuje metody souboru `command_parser.go`.

parseCommand()

Tato metoda bere jako vstup řetězec. Tento řetězec je pak transformován na pole argumentů, kde první položka pole je samotný příkaz a následující položky jsou pak parametry příkazu.

[1] https://go.dev/blog/slices-intro

LoadCommand()

Tato metoda bere jako vstup otevřený soubor a čeká na vstup od uživatele. Po přijmutí vstupu od uživatele smaže symbol nové řádky a využije metody `parseCommand()` k získání pole argumentů, který je pak návratovou hodnotou metody.

ParseFormatString()

Tato metoda je speciální metoda určená pro parametr příkazu `format`. Parametr příkazu je řetězec, který vždy obsahuje velikost souboru ve formátu `ČÍSLO;VELIKOST` (bez středníku). `VELIKOST` zde muže napsaná v jednotkách kilo, mega, giga či i tera. Metoda transformuje řetězec na číslo, které představuje velikost v bytech.

2.1.2 Základní struktury `fs_structs.go`

Tato sekce stručně popisuje základní struktury souboru `fs_structs.go`, která byla potřebná pro implementaci tohoto zadání.

Superblock

Struktura **Superblock** představuje zjednodušenou Superblock strukturu v systémech založených na i-uzlech. Tato struktura obsahuje záznamy jako jsou například:

- `Signature`: Podpis autora souborového systému.
- `VolumeDescriptor`: Popis souborového systému.
- `DiskSize`: Celková velikost virtuálního souborového systému.
- `ClusterSize`: Velikost clusteru.
- `ClusterCount`: Počet clusterů.
- `InodeCount`: Počet inode.
- `BitmapiStartAddress`: Adresa začátku bitmapy pro inode.
- `BitmapiSize`: Velikost bitmapy pro inode v bytech.
- `BitmapSize`: Velikost bitmapy pro data v bytech.
- `BitmapStartAddress`: Adresa začátku bitmapy pro datové bloky.
- `InodeStartAddress`: Adresa začátku inode.
- `DataStartAddress`: Adresa začátku datových bloků.

PseudoInode

Struktura **PseudoInode** představuje i-uzel. Počet přímých uzlů je nastaven na 12 a počet nepřímých je nastaven na 3, ale využívají se jenom 2.

- `NodeId`: ID inode.
- `IsDirectory`: Určuje, zda se jedná o soubor nebo adresář.
- `References`: Počet odkazů na inode, pokud počet referencí bude 0, indikuje to, že soubor má být smazán
- `FileSize`: Velikost souboru v bytech.
- `Direct`: Přímé odkazy na datové bloky.
- `Indirect`: Nepřímé odkazy na datové bloky.

SinglyIndirectBlock

Struktura **SinglyIndirectBlock** představuje blok, který obsahuje ukazatele na datové bloky.

- `Address`: Vlastní adresa bloku.
- `Pointers`: Pole ukazatelů na datové bloky.

DoublyIndirectBlock

Struktura **DoublyIndirectBlock** představuje blok, který obsahuje pole ukazatelů, kde každý ukazatel ukazuje na pole ukazatelů, které ukazují na datové bloky. Zjednodušene řečeno se v této implementaci toto reprezentuje jako pole struktury **SinglyIndirectBlock**.

- `Address`: Vlastní adresa bloku.
- `Pointers`: Pole struktur **SinglyIndirectBlock**.

DirectoryItem

Struktura **DirectoryItem** je jednoduchá struktura, která provádí přiřazení čísla i-uzlu na název položky.

- `Inode`: ID inode odpovídající souboru.
- `ItemName`: Název položky v adresáři.

2.1.3 Metody `command_interpreter.go`

Tato sekce popisuje nejdůležitější metody souboru `command_interpreter.go`. Podporované příkazy jsou format, incp, cat, ls, mkdir, cd, rmdir, rm, pwd, info, cp, mv, outcp, load, xcp, a short viz [1] pro popis těchto příkazu. Každý příkaz je implementován metodou se stejným názvem jako příkaz.

Je zde využívána konstrukce[2] programovacího jazyka Go, která podporuje přiřazování metod ke strukturám.

NewInterpreter()

Tato metoda představuje továrnu pro příkazové intepreta, kde vstupem je otevřený soubor. Vytvoří a vrátí nového příkazového interpreta s daným otevřeným souborem. Příkazový interpret by se měl vytvářet pomocí této metody pro správnou funkčnost aplikace.

ExecFormat()

Tato metoda bere jako vstup velikost virtuálního filesystému a název. Na uživatelovi OS se v pracovním adresáři vytvoří nový soubor s daným názvem a velikostí. Provede se inicializace Superblock a vytvoření hlavního adresáře. Pak v tomto souboru (filesystému) bude uložen Superblock, bitmapy datových bloků a i-uzlů a hlavní adresář.

ExecCommand()

Tato metoda bere jako vstup pole argumentů, kde první položka pole reprezentuje příkaz. Pomocí `switch` konstrukce se pro každý podporovaný příkaz spustí navazující metoda k obsloužení příkazu.

LoadInterpreter()

Provádí načítání `Superblock`, bitmap datových bloků, bitmap i-uzlů a momentálního pracovního adresáře z filesystému. Tato metoda by se měla volat po každým spuštěném příkazu pro správné fungování aplikace.

2.1.4 Metody `fs_commands.go`

Tato sekce popisuje nejdůležitější metody souboru `fs_commands.go`. Metody tohoto souboru provádí veškerou manipulaci s virtuálním filesystémem.

[2] https://gobyexample.com/methods

GetAvailableDataBlocks()

Tato metoda vezme byte `slice` dat, zpočítá kolik datových bloků je potřeba a vrátí seznam adres datových bloků s novou modifikovanou bitmapu, která představuje bitmapu po zabrání dat. Tato metoda nemodifikuje vstupní bitmapu.

GetAvailableInodeAddress()

Tato metoda prohledá bitmapu i-uzlů, najde volný i-uzel a následně vrátí adresu i-uzlu a novou bitmapu, která představuje bitmap i-uzlů po zabrání daného volného i-uzlu. Tato metoda nemodifikuje vstupní bitmapu.

WriteAndSaveData()

Tato metoda vezme data a vytvoří nový i-uzel reprezentující nový soubor s těmito daty, který je pak návratovou hodnotou metody. Data, i-uzel a upravené bitmapy jsou uloženy do filesystému.

GetFileClusters()

Tato metoda vezme i-uzel souboru a prohledá veškeré datové bloky tohoto souboru. Následně metoda vrací seznam adres datových bloků souboru a také vrací datové bloky, které byly alokovány k uchování nepřímých a dvojitých nepřímých odkazů.

2.1.5 Metody `main.go`

Tento soubor obsahuje pouze metodu `main()`. Tato metoda je hlavní vstupní bod do aplikace. Aplikace vyžaduje při spuštění jeden parametr. Tento parametr představuje název filesystému. Pokud již filesystém existuje, automaticky se načte a uživatel může začít zadávat podporované příkazy. Pokud neexistuje, čeká se dokud uživatel nevykoná příkaz `format`. Pokud uživatel použije jiný příkaz, zatímco filesystém ještě nebyl naformátován, aplikace vypíše chybovou hlášku a ukončí se.

Kapitola 3

Uživatelská příručka

3.1 Přeložení a sestavení programu

Aplikace byla vyvinuta v programovacím jazyce Go verze 1.21.5. Je tudíž potřeba nainstalovat tento jazyk na počítač viz `https://go.dev/`.

3.2 Použití programu

Program by se měl spouštět příkazem:

`go run .` ⟨`nazev—filesystemu`⟩

Symbol ⟨`nazev-filesystemu`⟩ představuje nutný argument aplikace, což je název filesystému. Pokud filesystém je již už vytvořen, automaticky se načte a uživatel může zadávat podporované příkazy. Pokud filesystém neexistuje na uživatelském OS, aplikace čeká až uživatel zadá příkaz:

`format` ⟨`cislo`⟩⟨`velikost`⟩

Symbol ⟨`cislo`⟩ zde představuje číslo. Symbol ⟨`velikost`⟩ zde představuje jeden ze znaků "", "K", "M", "G", "T", kde prázdný znak představuje velikost v bytech a ostatní znaky představují jednotky kilo, mega, giga, tera.

Následně může uživatel začít zadávat podporované příkazy (popsané viz [1]) jako jsou:

`format`, `incp`, `cat`, `ls`, `mkdir`, `cd`, `rmdir`,
`rm`, `pwd`, `info`, `cp`, `mv`, `outcp`, `load`, `xcp`, `short`

Literatura

[1] Ladislav Pešička. ZADÁNÍ SEMESTRÁLNÍ PRÁCE - Semestrální práce ZOS 2023 (verze dokumentu 01). `https://courseware.zcu.cz/portal/studium/courseware/kiv/zos/samostatna-prace.html`, 2023. [Online].