FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Semestrální práce

Souborový systém založený na i-uzlech

Jakub Vokoun

PLZEŇ

2025

Obsah

Zadání 1

Vytvořte zjednodušený ***souborový systém*** založený na ***i-uzlech***. Implementujte jednoduché ***TUI*** pro práci se souborovým systémem které umožní uživateli používat konkrétní příkazy s definovaným chováním. Bonusový úkol pro studenta s orion loginem začínajícím na písmeno **j**.

Kompletní znění zadání včetně popisu všech příkazů je na straně: 6.

Popis implementace 2

Program je napsán v ***c++ 23*** s využitím standardní knihovny a knihovny `readline` která slouží k uchování historie příkazů za běhu programu.

Kompletní implementace souborového systému se nachází ve třídě `Filesystem`. Tato třída nabízí metody pro práci se systémem, jak elementární funkčnost (například `inode_create`) tak komplexnější (například `file_create`), které vhodně spojují funkčnost elementárních funkcí do uceleného celku metody vyšší úrovně abstrakce.

Při změně velikosti souboru (zvětšování, jelikož zmenšování podle zadaných příkazů není třeba, kromě zmenšení na 0, tedy mazání souboru) je zajištěna atomicita všech operací, jelikož se jedná o častou operaci, která má velkou šanci neúspěchu. Využívá se rozdělení klastrů na režijní a datové. Datové jsou ty, které přímo obsahují data souboru, zatímco režijní jsou nepřímé prvního a druhého řádu, obsahující odkazy na datové respektive režijní klastry. Nejprve se předalokuje dostatek klastrů jak pro data tak režii - tím se eliminuje většina problémů pramenících z nedostatku volného místa. Následuje zápis režijních klastrů prvního řádu do klastrů druhého řádu, po čemž se zapíší datové klastry do režijních prvního řádu. Až po provedení těchto operací se zapíše nová velikost do i-uzlu. Konzistence dat je zajištěna dopředným načtením všech existujících klastrů a jejich zápis ve stejném pořadí.

Všechny příkazy které se dají v TUI použít mají vlastní třídu, které je oddědena od třídy `ICommand` s virtuální metodou `execute_inner` a společnou metodou `execute`, která volá virtuální, odchytává jakékoliv ***výjimky*** a vypisuje standardní zprávy definované v zadání, popisující úspěch nebo chybu při běhu příkazu.

Uživatelská příručka 3

Program je třeba zkompilovat a to buď pomocí standardních `cmake` příkazů, anebo přiloženého ***bash skriptu*** `auto/build.bash`, který vhodně aplikuje standardní cmake na zvolenou adresářovou strukturu. Pro kompilaci nejspíš lze použít gcc, autorem byl použit `clang 20.1.8`.

Spustitelný soubor lze spusit standardně, bude se ale nacházet v adresáři `bin`. Pro jednoduchost lze použít přiložený skript `auto/run.bash`, který se zároveň postará o zadání jména souboru, ve kterém se souborový systém nachází.

Při spuštění lze specifikovat příznak `-v` nebo `--vocal`, které při běhu programu vypisují další diagnostické informace.

Po spuštění programu lze zadávat všechny příkazy specifikované zadáním. Navíc byly přidány dva další: `help`, který vypisuje seznam všech použitelných příkazů a `exec`, který umožňuje spouštět ***shell*** příkazy. Tento příkaz byl přidán z důvodu snadného testování.

Každý příkaz lze zavolat s příznakem `-h` po kterém se místo provedení příkazu zobrazí nápověda k jeho používání.

Závěr 4

Souborový systém je ve funkčním stavu s uspokojivou úrovní funčnosti. Před rozšiřováním funkcionality by bylo záhodno se zaměřit na fungování třídy `Filesystem`, konkrétně na:

- použití návrhového vzoru ***jedináček***. Toto rozhodnutí výrazně urychlilo vývoj, ale zhodnotit užitečnost tohoto rozhodnutí je zcela na místě.
- dekompozici třídy. Ta jako taková zastává více povinností, než by bylo moudré. Dekomponovat by šlo horizontálně (rozdělit na třídy, které se zaměřují pouze na práci s i-uzly, klastry,...) anebo vertikálně (rozdělit na nízkourovňovou funkcionalitu, tj. práci s i-uzly, klastry, a vysokoúrovňovou, tj. práci se soubory a adresáři).
- práci s režijními klastry - unifikovat logiku tak, aby bylo jednoduché přidávat nepřímé režijní klastry vyššího řádu, bez nutné úpravy kódu.

Kromě této hlavní třídy by měl být zrevidován způsob emulování terminálu, přesunutí funkcionality do třídy ze souboru `main.cpp`.

Kompletní zadání

Následující strany obasahují kompletní a nezkrácené zadání semestrální práce.

Semestrální práce ZOS 2025 (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, můžete si však přidat další pomocná hlášení, jakou jsou podrobnosti k chybám či doplňující vypisované informace.

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, nebo přejmenuje s1 na 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
ls
```

Možný výsledek:

```
FILE: f1
DIR: a2
```

```
PATH NOT FOUND (neexistující adresář)
```

7) Vypíše obsah textového souboru s1 na obrazovku

```
cat s1
```

Možný výsledek:

```
OBSAH TEXTOVÉHO SOUBORU
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
info s1
```

Možný výsledek:

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

```
FILE NOT FOUND (není zdroj)
Může vypadat např. takto u clusteru velikost 1024B:
 ahoj.txt – 1800 B – i-uzel 7 – přímé odkazy 101, 102
```

11) Nahraje soubor s1 z pevného disku do umístění s2 ve vašem FS

```
incp s1 s2
```

Možný výsledek:

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

12) Nahraje soubor s1 z vašeho FS 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
```

15) Příkaz exit – ukončení práce s vaším programem.

16) Příkaz statfs – vypíše statistiky souborového systému, jako je velikost, počet obsazených a volných bloků, počet obsazených a volných i-uzlů, počet adresářů.

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:

- Hardlink (příkaz *ln s1 s2*) – pokud login studenta začíná **a-i**
  Vytvoří hard link na soubor s1 s názvem s2. Dále se s ním pracuje očekávaným způsobem, tedy např. cat s2 vypíše stejný obsah jako cat s1. Také dojde k úpravě příkazu info, aby zobrazoval i počet hardlinků odkazujících na daná data.

- Manipulace se soubory – pokud login studenta začíná **j-r**
  příkaz *xcp s1 s2 s3* - Vytvoří soubor s3, který bude spojením souborů s1 a s2.
  příkaz *add s1 s2* – Přidá na konec souboru s1 obsah souboru s2

- 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. Také dojde k úpravě příkazů ls a info, kde uvidíme, že daný soubor je symbolickým linkem.

Odevzdání práce

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

Osobní předvedení práce cvičícímu. Referenčním strojem je školní PC v UC326 nebo UC329.
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 2026).
- 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
- Kvalita dokumentace