FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV/ZOS

SEMESTRÁLNÍ PRÁCE 2017 PSEUDOFAT

Martin Hamet
A14B0254P

9.února 2017

Obsah

1 Zadání

Vytvoření pseudoFAT souborového systému, splňujícího následující úlohy.

- Nahraje soubor z adresáře do cesty virtuální FAT.
- Smaže soubor z virtuální FAT.
- Vypíše čísla clusterů, oddělené čárkou, obsahující data souboru.
- Vytvoří nový adresář v zadané cestě.
- Smaže prázdný adresář v zadané cestě.
- Vypíše obsah souboru na obrazovku.
- Vypíše strukturu adresářové struktury včetně souborů.
- Zkontroluje, zda je souborový systém nepoškozen. V případě poškození zobrazí nalezené chyby a opraví je.

Omezení:

- Maximální počet položek v adresáři bude omezen velikostí 1 clusteru
- Maximální délka názvu souboru bude 8+3=12 znaků (jméno.přípona) + ukončovací znak, tedy 13 bytů.
- Každý název bude zabírat právě 12 bytů (do délky 13 bytů doplníte ukončovacím znakem - při kratších názvech).
- Každý adresář bude zabírat přesně 1 cluster (i když nebude plně obsazen).
- Protože se bude pracovat s velkým objemem dat, je potřeba danou úlohu paralelizovat.

2 Analýza problému a návrh řešení

Úkolem je vytvoření pseudoFAT souborového systému. Aplikace není zamýšlena pro praktické využívání, proto není nutné zabývat se datovou kapacitou souborového systému, což usnadní ladění. Souborový systém by měl obsahovat informace o typu velikostech atp. dále pomocné struktury které budou podporovat logiku funkce a vlastní uložená data.

2.1 Souborový systém FAT

Budeme se držet základního principu funkce souborových systémů FAT. Tedy systém se skládá ze dvou hlavních prvků, tabulka referencí (FAT tabulka) a clusterů. Obě tyto části mohou být snadno reprezentované polem o stejné velikosti.

2.1.1 Cluster

Cluster je ucelený paměťový blok, pro jednoduchost pevné velikosti. Každému clusteru odpovídá jedna adresa, tedy není možné mít více clusterů než je schopné adresovat. Pro uložení dat do souborového systému se vyhradí nejmenší větší počet celých clusterů, který obsáhne daná data. Z čehož plyne problém nevhodného zvolení velikosti clusterů. Pokud bude velikost clusteru příliš malá bude potřeba vetší adresovací schopnost pro využití stejného místa. Naopak při velké velikosti clusteru se plýtvá místem. V případě souboru menšího než je velikost clusteru soubor bude fyzicky zabírat celý cluster jakožto nejmenší adresovatelnou jednotku dat.

2.1.2 FAT tabulka

Jedná se o tabulku referencí na clustery. Jeden záznam v tabulce odpovídá jednomu clusteru. Funkce tabulky spočívá v poskytování informací o stavu clusteru (volný, konec souboru, adresář, poškozený).

Příklad uloženého souboru s obsahem `Obsah textového souboru.`

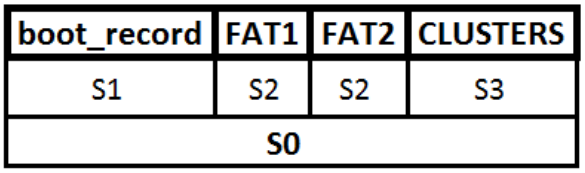
`Index`	`FAT tabulka`	`Obsah clusteru`
5	6	*Obsah*
6	9	*textovho*
7	FAT_UNUSED	$f4fsd65faf8$
8	$FAT\_DIRECTORY$	$fad5sf16af1$
9	FAT_END_FILE	*souboru.*

Obrázek 1: Příklad uložení souboru.

V praxi by jsme byli omezeni počtem indexů (adres). V případě číselného typu `uint8` (8 bytový integer) se jedná o maximální hodnotu 255. Z toho plyne pokud by jsme tvořili souborový systém s takovou adresovatelností mohli by jsme použít pouze 255 clusterů. Pro zaznamenání stavu clusteru budeme potřebovat 4 zmíněné hodnoty. Tedy počet adresovatelných clusterů se sníží o 4 (251 hodnot). Návaznost clusterů (např. několik clusterů obsahujících data souboru) se v tabulce projeví jako odkaz na další index clusteru v pořadí viz Obr. 1.

2.2 Struktura souborového systému

Souborovým systémem bude v našem případě myšlen výše zmíněný soubor `.dat`. V praxi takový soubor nemůže být libovolně velký a má pevně danou velikost. Tato skutečnost omezuje počet (i velikost) clusterů a tabulky. Samotné clustery jsou víceméně nepoužitelné bez příslušné tabulky, proto se tabulka v souborovém systému vyskytuje alespoň ve dvou kopiích, kdyby došlo k poškození. To znamená že je se kapacita souborového systému dále sníží viz Obr. 2. Na obrázku `S0` znázorňuje celkovou velikost souborového systému. Oblast použitelná pro data je `CLUSTERS`. Z celkové velikosti musíme odečíst velikost záznamu o souborovém systému `S1`, dvakrát velikost tabulky FAT `S2`, která je daná velikostí integeru bez 4 (konstanty pro stavy clusterů viz výše) a tím získáme velikost `S3`. Z velikosti clusterů (`S3`) snadno zjistíme velikost jednoho clusteru, vydělením počtem záznamů FAT tabulky.



Obrázek 2: Struktura souborového systému.

Pro případ naší simulace nebudeme uvažovat omezení velikosti souborového systému `S0` a zvolíme velikost clusteru 256 bytů a počet použitelných clusterů bude 251. Neuvažujeme tedy velikosti `S1, S2`.

2.2.1 Soubor

Soubor je tvořen řetězem referencí ve FAT tabulce viz Obr. 1, vlastními daty v odpovídajících clusterech a svým záznamem v příslušném adresáři (viz 2.2.2). Záznam v adresáři určí počáteční cluster a následující clustery lze vyčíst z FAT tabulky. Poslední soubor clusteru bude označený v tabulce jako `FAT_FILE_END`.

2.2.2 Adresář

Adresář ve své nejjednodušší podobě funguje jako uzel shromažďující informace o složkách a souborech které mu jsou přiřazeny. Neobsahuje přímo jejich data ale pouze záznamy o nich. Jeho umístění v souborovém systému vyřešíme tak že vyhradíme jeden celý cluster, který ve FAT tabulce bude označen jako adresář (`FAT_DIRECTORY`). Tento cluster bude sloužit pouze pro struktury adresáře. Adresářový cluster budeme plnit jednotlivými záznamy o souborech a dalších adresářích které obsahuje. Konkrétně viz Obr. 3.

Záznam	Jméno	Soubor	Velikost	Počátek
1	$soubor1$	$True$	$100B$	2
2	$soubor2$	$True$	$200B$	5
3	$sloka1$	$False$	0	8
4	$sloka2$	$False$	0	9

Obrázek 3: Příklad adresářového clusteru.

V každém záznamu je uvedeno zda se jedná o záznam souboru (`True`) nebo dalšího vnořeného adresáře (`False`). V případě souboru i jak je soubor velký. Toho využijeme při čtení souboru, protože máme pouze informaci o jemu přiřazených clusterech. Díky velikosti zjistíme kolik z místa soubor zabírá ve svém posledním clusteru. V obou případech záznam obsahuje počátek, který určuje index clusteru, kde začínají data daného záznamu. Díky této struktuře bude možné rozvětvovat strom adresářů pouze s omezením počtu volných clusterů.

Záznam v adresářovém clusteru bude zabírat pevně danou velikost. Z toho plyne omezení počtu záznamů na adresář. Jinak by adresář přesahoval svůj přidělený cluster (počátek) a bylo by nutné situaci řešit jinak.

Aby bylo možné vkládat soubory a složky bude v základu vytvořen adresář `root` na prvním clusteru ze kterého se bude vždy vycházet.

2.3 Informace o souborovém systému

Podle vzoru bude vhodné zavést strukturu `boot record` která by měla obsahovat veškeré základní informace o souborovém systému (popis, typ, počet fat tabulek, velikost clusteru, počet použitelných clusterů). Tento záznam by se měl vyskytnout na začátku souboru `.dat` (souborového systému).

2.4 Kontrola souborového systému

Snaha je o udržení souborového systému v co nejlepším stavu a především zabránit ztrátě dat. Je nutné tedy kontrolovat konzistenci souborů a záznamů o nich podobně jako adresářů. Systémovým chybám lze předejít např. logováním (tvoření žurnálu o změnách). V naší simulaci se ale zaměřím spíše na opravu již vzniklých chyb.

2.4.1 Hledání chyb

Pro hledání chyb lze použít různé metody v této práci sem zvolil pouze několik základních principů.

hit test Jako hlavní metodu odhalování chyb jsem zvolil hit test. Nejprve bude nutné vytvořit pole čítačů (pro každý cluster jeden čítač). Metoda spočívá v rekurzivním průchodu adresářové struktury z kořenového adresáře a zvýšení příslušného čítače při objevení záznamu (adresář, soubor). U souboru je nutné zvýšit čítač u každého clusteru který je mu přidělený. Výsledkem je pole znázorňující kolikrát byl každý cluster nalezen. Porovnáním tohoto výsledku a záznamů ve FAT tabulce můžeme vyvozovat různé chyby viz 2.4.2.

vlastnosti souboru Je možné odhalit neukončený soubor podle jeho záznamu o velikosti nebo tvaru jména (duplicita, délka). Pokud velikost všech přidělených clusterů přesahuje velikost v záznamu souboru. Případně pokud by soubor jako svůj další cluster odkazoval na adresář nebo na nevyužitý cluster.

sekundární FAT tabulka Při rozdílných záznamech v primární a sekundární FAT tabulce. Bohužel se již dále bude hůře určovat, která tabulka je chybná proto se budu spíše spoléhal na tabulku primární a v případě podezření na chybu kontrolovat s tabulkou sekundární. Je značně nižší šance že se náhodná chyba projeví v obou tabulkách. Při neshodě tabulek by mohlo být užitečné soubor ukončit a pokusit se z clusterů bez reference vytvořit samostatný soubor.

2.4.2 Oprava chyb

Při opravování chyb nelze vždy určit správný postup a rozhodování není triviální proto se budeme držet jednodušších mechanismů.

Za nejužitečnější nástroj bych považoval výše zmíněný hit test.

Případy hit testu:

- **odkaz na nepoužitý cluster** Předpokládáme správnost odkazu a pokusíme se podle typu odkazu obnovit adresář nebo soubor. V případě obnovení adresáře hrozí že se obnoví množství neplatných souborů pokud bude obsahovat platné adresy. Tento problém se nedá předem odhalit a proto bych se o obnovu radši pokusil. Aby nedocházelo k takovému stavu často proběhne nejprve kontrola obou tabulek, a obnovení proběhne pouze v případě, že obě tabulky obsahují stejný záznam. Tedy v obou je daný cluster označen jako např. jako adresář.

- **násobný odkaz na adresář** V běžném souborovém systému by to nebyla chyba (linkování atp.), v našem případě ale duplicitní odkaz odstraníme. Zabráníme tím násobnému opravování souborů ve vnořených adresářích. Podobně jako u datového clusteru bude rozhodnutí o znovuzavedení adresáře záviset na shodě FAT tabulek.

- **adresář bez reference** Ztráta reference na adresář by se měla vždy řešit pokusem o obnovu. Vytvoření nového adresářového záznamu s odkazem na nalezený cluster v kořenovém adresáři. Tato operace vyžaduje nový hit test z toho důvodu, že adresář mohl obsahovat další záznamy a bude lepší je hned zahrnout do testování než se je pokoušet obnovit "na slepo".

- **násobná reference na cluster** Jedná se pravděpodobně o křížení souborů. Aby nedošlo k poškození ani jednoho z nich (pokud je některý v pořádku) bylo by vhodné najít soubory, kterým zmíněné clustery patří a vytvořit kopii. Jeden soubor zkopírovat na nové místo, aby se při jejich další změně nepřepisovali data obou. Vzhledem k tomu že v naší simulaci není možné uložené soubory přepisovat tak tuto opravu pominu z důvodu komplikovaného rozhodování v kombinaci s ostatními chybami. Křížení budeme řešit pomocí záznamu hit testu a pokud by nastala "potkání"clusteru který už byl zaznamenán v hit testu ukončíme právě zkoumaný soubor.

- **datový cluster bez reference** Mohlo by se jednat o soubor na který se ztratila reference. Pokračováním podle návaznosti dle FAT tabulky by mohlo zapříčinit "roztrhání"souboru na několik nalezených. Proto bude lepší řešit tyto případy na konci kontroly, kdy už víme o všech clusterech tohoto typu. Z množiny těchto clusterů se pokusíme podle FAT tabulky sestavit co nejdelší řetězce vedoucí ke konci souboru. Těmto řetězcům dále vytvořit záznamy v kořenovém adresáři.

Ostatní případy:

- **záznam o souboru odkazuje na nevalidní cluster** Naše struktura neobsahuje zálohy adresářů. Není tedy kde začít a záznam o souboru bude smazán.
- **soubor v průběhu obsahuje odkaz na nevalidní cluster** V případě že nevalidní cluster je adresář bude soubor ukončen záznamem `FAT_FILE_END` na svém předchozím clusteru aby nedošlo k poškození dalších souborů. V ostatních případech je možné zkontrolovat stav se sekundární FAT tabulkou a nahradit záznam v primární, jinak bude soubor ukončen.
- **počet clusterů souboru neodpovídá jeho velikosti** Soubor bude ukončen primárně podle své velikosti aby se zabránilo poškozování dalších souborů.
- **dva záznamy ve stejném adresáři mají stejný název** Prvnímu takovému záznamu s duplicitou bude vygenerováno nové jméno.
- **název záznamu neodpovídá standardu** Standardní velikosti jména je 12 znaků včetně tečky oddělující třípísmennou koncovku. Nestandardnímu souboru bude vygenerováno nové jméno s ohledem na duplicitu.

3 Implementace

Program je připraven na případnou změnu na jinak velký souborový systém nicméně některé součásti jsou závislé na zvoleném typu souborového systému podle vzoru FAT8.

3.0.1 Hlavní části programu

`main` Zajišťuje zpracování předaných parametrů načtení (`load_fat`) a uložení (`save_fat`) souborového systému a volání příslušných funkcí podle přepínačů.

`import_file` Zajistí načtení souboru (`load_file`) z disku vytvoření příslušného záznamu (`create_file_rec`) podle zadané adresy a uložení do souborového systému.

`create_directory` Zajistí vytvoření adresáře a jeho záznamu (`create_directory_rec`) podle zadaných parametrů.

`get_directory_cluster` Slouží ke zpracování zadané adresy do souborového systému a získání příslušného indexu adresářového clusteru.

`get_record` Z adresy do souborového systému získá ukazatel přímo na hledaný záznam (`get_directory_cluster`), (`get_node_offset`).

`check_structure` Zajišťuje hlavní kontrolu souborového systému (`hit_test_arr`). Poskytuje obecné informace o kontrole. V případě potřeby spouští obnovu nalezených souborů a adresářů(`recover_directory`), (`file_recovery`).

`hit_test_arr` Rekurzivně prochází adresářovou strukturu a zaznamenává nalezené soubory a složky (`mark_directory`), (`mark_file`) do záznamu hit testu. A provádí vedlejší kontroly (jméno, velikost, validní odkazy atp.) a příslušné opravy.

`file_recovery` Po dokončení ostatních oprav prochází finální výsledky hit testu a tvoří list zřetězených clusterů které k sobě patří (dle FAT tabulky referencí). Po dokončení průchodů založí záznamy, pro každý řetězec (potencionální soubor) v kořenovém adresáři.

4 Uživatelská příručka

Program je určen pro spouštění z příkazové řádky. Podle zadaných parametrů vykonává požadované funkce nad souborem `.dat`.

4.0.1 Ovládání programu (přepínače)

`ZOS_FAT.exe soubor.dat -a S1 ADR` Nahraje soubor z adresáře do cesty virtuální FAT tabulky.
S1 soubor pro nahrání
ADR adresa do souborového systému

`ZOS_FAT.exe soubor.dat -f S1` Smaže soubor S1 ze soubor.dat.
S1 plná cesta k souboru

`ZOS_FAT.exe soubor.dat -c S1` Vypíše čísla clusterů, oddělené dvojtečkou, obsahující data souboru S1.
S1 plná cesta k souboru

`ZOS_FAT.exe soubor.dat -m ADR ADR2` Vytvoří nový adresář ADR v cestě ADR2.
ADR název nového adresáře
ADR2 cesta pro nový adresář

`ZOS_FAT.exe soubor.dat -r ADR` Smaže prázdný adresář ADR.
ADR cesta k adresáři

`ZOS_FAT.exe soubor.dat -l S1` Vypíše obsah souboru S1 na obrazovku.
S1 plná cesta k souboru

`ZOS_FAT.exe soubor.dat -p` Vypíše adresářovou strukturu s odpovídajícím odsazením.

`ZOS_FAT.exe soubor.dat -s` Provede kontrolu, případné opravy a výpis výsledku.

`ZOS_FAT.exe soubor.dat -e` Vytvoří nový soubor empty.dat s prázdným souborovým systémem.

5 Závěr

Parametry souborového systému byli zvolené tak, aby bylo snazší ladění což se projevilo jako výhoda. Jednotlivé úkoly nebyly náročné na implementaci po připravení základní funkčností systému. Naopak značné problémy způsobovala kontrola a oprava souborového systému. Opravy i kontrola by šla výrazně vylepšit, ale náročnost hledání a opravování chyb rychle roste. S novými způsoby oprav přibývají možnosti poškození ostatních struktur a je třeba komplexnější rozhodování. Přesto se podařilo připravit program na detekci a opravu jednoduchých chyb. Program je schopen obnovit ztracené adresáře, soubory a srovnat nebo částečně opravit neukončené soubory. Přerušené soubory je program schopen detekovat a obnovit alespoň jako dva nezávislé.

Paralelizace načítání a ukládání souboru se po hlubším zkoumání projevila jako téměř zbytečná, při uvážení plnění bufferu jedním vláknem a zpracování dat z bufferu ostatními, protože místo bufferu může vlákno ukládat data rovnou na požadované místo. Paralelní zpracování by tedy přicházelo v úvahu pro kontrolu souborového systému což se projevilo jako poměrně náročný úkol a z časových důvodů jsem tuto variantu zavrhl.