Západočeská univerzita
Fakulta aplikovaných věd

Semestrální práce
Základy operačních systémů

File Allocation Table

Jakub Váverka

23. ledna 2017

Email: vaverkaj@students.zcu.cz
Git: https://github.com/LaserPork/ZOS.git

FAT hodnoty		
Typ	Výpočet	Hodnota
FAT_UNUSED	INT32_MAX - 1	2147483646
FAT_FILE_END	INT32_MAX - 2	2147483645
FAT_BAD_CLUSTER	INT32_MAX - 3	2147483644
FAT_DIRECTORY	INT32_MAX - 4	2147483643

1 Úvod

Účelem této práce je bližší seznámení s tabulkou FAT. FAT je souborový systém, který využívá stejnojmennou tabulku pro ukládání informací o souborech a složkách, které jsou uloženy na disku. V této práci nedochází k vytváření reálného disku, ale spíše simulaci na souboru, který má podobnou datovou strukturu. Ke zpracování jsem použil programovací jazyk C.

2 Zadání

viz. https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=125239

3 Řešení

Pro řešení jsou použil stejné datové struktury jako byly v ukázkovém příkladu na https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=130007

Soubor který simuluje FAT se skládá pouze z 2 kopií tabulek FAT a jednotlivých clusterů. Vynechal jsem tedy počáteční boot_record, který se hodí jen v případě, že nevíme jak je FAT tabulka vytvářena. Velikost tabulek FAT je 251. Což udává i maximální počet clusterů. Pokud je cluster prázdný, konec souboru, špatný a nebo složka, je jeho hodnota určena jako $int32_t MAX - n$.

Cluster samotný má velikost 256B. Obsah který je v něm uložený musí být ukončen '\0', takže konečný počet znaků, které můžeme z clusteru využít je jen 255. Pokud je tento cluster označen jako složka, může obsahovat soubory nebo podsložky. Ty jsou reprezentovány strukturami *directory*. V této struktuře je uložen název, který je dlouhý 12 znaků + ukončovací nula. Můžeme tady ukládat soubory s názvem dlouhým 8 znaků + 1 znak tečka + 3 znaky koncovka. Např. *mujsoubo.txt*. Dále je v této struktuře uložena pravdivostní hodnota, zda se jedná o složku či adresář, velikost daného souboru (u složky je rovna nule) a číslo clusteru na kterém daný soubor nebo

složka začíná. Tato struktura má 24B. Z toho se dá již určit, že maximální počet souborů a složek v jedné složce je 10 (256/24 zaokrouhleno dolů).

Běh Pokud je program spuštěn bez parametrů, zobrazí nápovědu. Argumenty na které program reaguje, jsou definovány v zadání. Navíc program reaguje na 3 argumenty vlastní. Pokud je spuštěn s argumentem $-dump$, vypíše na obrazovku seznam všech clusterů, které nejsou prázdné. Tento příkaz slouží pouze pro rychlou kontrolu. Další vlastní příkaz je $-corrupt$. Ten slouží k umělému vytvoření badbloků. Projde všechny clustery a pokud je cluster součástí souboru, nahradí prostřední znaky clusteru označením badbloku (FFFFFF). Posledním příkazem je $-repair\ threadCount$. Tento příkaz projde FAT a opraví veškeré badbloky. Za parametr threadCount, je dosazen počet vláken, které mají tuto operaci provádět. Při spuštění programu dojde k načtení souboru FAT, pokud na dané cestě žádný není, je vytvořen prázdný. Poté se provede příkaz zadaný argumenty a změněná data se znovu do souboru uloží.

Paralelní zpracování Pro paralelní zpracování jsem použil linux knihovnu pthreads. Před spuštěním opravy se vytvoří pole mutexů, každý odpovídá jednomu clusteru. Vlákna začínají na prvním clusteru. První vlákno zkontroluje jestli není cluster uzamčený, pokud není, zjistí jestli je špatný. Pokud ano, tak uzamkne příslušný mutex, najde prázdný cluster, uzamkne i jeho mutex a začne kopírovat. Každé vlákno vlastní zámek nad maximálně 2 mutexy. Po ukončení práce odemkne mutexy a přejde k dalšímu clusteru.

4 Překlad

Program byl přeložen a testován na unixovém operačním systému Debian verze Jessie. Překlad lze provést pomocí přiloženého cmake a nebo tímto příkazem. $gcc - oprogrammain.c - lpthreads - lm$ Pro překlad jsou zapotřebí knihovny pthreads.h a math.h.

5 Závěr

Uvědomuji si, že můj přístup není úplně ideální. Chtěl jsem ho po prvních testech zoptimalizovat. Poté co jsem svou aplikaci otestoval, jsem si však uvědomil že je velikost dat na testování moc malá. K opravě dochází v řádech milisekund a rozdíly mezi vícevláknovým a lineárním přístupem jsou neměřitelné. Nicméně si myslím, že smysl práce byl naplněn. Na vlastní kůži

jsem si vyzkoušel jak funguje tento jednoduchý souborový systém a oprášil své znalosti programovacího jazyka C.