Souborový systém FAT32

9

Souborové systémy tvoří důležitou část operačních systémů. Ve srovnání s jinými částmi operačního systému máme v případě souborových systémů tu výhodu, že si jejich implementaci můžeme vyzkoušet pohodlně v uživatelském prostoru a v libovolném vhodném programovacím jazyce.[1] Náročnost implementace souborového systému se liší v závislosti na poskytovaných funkcích a použitých optimalizacích. V tomto cvičení si ukážeme základní strukturu a práci se souborovým systémem FAT32. Tento souborový systém je zajímavý jednak tím, že se s ním dá stále setkat na řadě míst, a hlavně tím, že je opravdu jednoduchý, jak by mělo být vidět z následujích řádek.[2]

1 Struktura souborového systému

Existuje několik verzí souborového systému FAT, které jsou označovány jako FAT12, FAT16 a FAT32.[3] Tyto verze mají identickou strukturu souborového systému, ale liší se formátem dat, zejména velikostí záznamů v tabulce FAT.[4] Obecnou strukturu souborového systému ukazuje následující schéma.[5]

rezervovaná oblast	tabulka FAT	kořenový adresář	oblast s daty
(boot sektor, informace FS)	(může být i kopie)	FAT12, FAT16 (u FAT32 možnost)	...

Která verze souborového systému bude použita, určuje velikost disku,[6] kde bude souborový systém uložen, přesněji řečeno, počet datových clusterů. Pro malé disky (s méně než 4085 clustery) se použije FAT12, pro větší disky (s méně než 65525 clustery) se použije FAT16 a pro (zbývající) velké disky se použije FAT32. Z toho plyne, že pokud bychom chtěli implementovat opravdu plnohodnotný souborový systém FAT, museli bychom implementovat všechny tři verze současně. V následujícím textu si práci usnadníme a budeme implementovat jen verzi FAT32 a uvedené informace se budou vztahovat pouze k této verzi. Implementace zbývajících verzí by byla velmi podobná, avšak místy bychom mohli narazit na drobné odlišnosti.

[1] Protože se jazyk C běžně používá pro vývoj operačních systémů, bude to naše první volba, ale klidně bychom mohli použít jazyk Java nebo Python.

[2] První verze souborového systému FAT se používaly na počítačích, které měly řádově *desítky* (maximálně nižší stovky) *kilobytů* RAM a podobně jako zbytek operačního systému MS-DOS byl souborový systém kompletně napsán v assembleru procesoru 8086. To je také důvod, proč souborový systém FAT obsahuje jen ty opravdu nejnutnější věci a z dnešního pohledu má řadu vážných nedostatků.

[3] Ještě existuje nástavba označovaná jako Virtual FAT (nebo VFAT), která je kompatibilní se souborovým systémem FAT a přidává podporu dlouhých jmen souborů.

[4] Budeme rozlišovat *souborový systém FAT* (jako celek) a *tabulku FAT* (strukturu nesoucí informace o umístění souborů).

[5] Více informací např. `https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system` nebo `https://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc`

[6] V tomto textu budeme používat pojem disk, i když souborový systém může být umístěn i jinde, např. v oddílu disku.

1.1 Boot sektor

První sektor disku obsahuje (mimo jiné) podrobné informace o vlastnostech souborového systému. Nás budou zajímat ty, které přesně definují jeho strukturu, např. velikost clusteru. Tabulka 1 představuje informace, které budeme potřebovat pro čtení ze souborového systému a pochází (včetně značení) z oficiální dokumentace.[7]

označení	offset (B)	velikost (B)	popis
`BPB_BytsPerSec`	11	2	velikost sektoru (v bytech); typicky 512
`BPB_SecPerClus`	13	1	počet sektorů v jednom clusteru (musí být mocnina 2)
`BPB_RsvdSecCnt`	14	2	počet sektorů rezervovaných na začátku disku
`BPB_NumFATs`	16	1	počet FAT tabulek (typicky 2)
`BPB_TotSec32`	32	4	celkový počet sektorů na disku
`BPB_FATSz32`	36	4	počet sektorů, které zabírá jedna FAT tabulka
`BPB_ExtFlags`	40	2	příznaky udávající způsob práce s kopiemi FAT tabulky
`BPB_FSVer`	42	2	upřesnění verze souborového systému
`BPB_RootClus`	44	4	číslo prvního clusteru kořenového adresáře
`BPB_FSInfo`	48	2	číslo sektoru v rezervované oblasti, kde jsou uloženy další informace souborového systému
`BS_BootSig`	66	1	příznak, že další 3 informace jsou přítomny
`BS_VolID`	67	4	pseudo-unikátní identifikátor média
`BS_VolLab`	71	11	popisek disku
`BS_FilSysType`	82	8	vždy obsahuje řetězec `"FAT32   "`

Tabulka 1: Vybrané informace uložené v bootsectoru

1.2 Adresáře

Metadata jednotlivých souborů (včetně adresářů) jsou uložena jako záznamy v adresářích. Tyto záznamy mají pevnou velikost 32 B a jejich strukturu popisuje Tabulka 2.

označení	offset (B)	velikost (B)	popis
`name`	0	11	název souboru
`attr`	11	1	atributy souboru
`ntRes`	12	1	rezervováno pro Windows NT
`crtTimeTenth`	13	1	čas vytvoření souboru
`fstClustHI`	20	2	číslo prvního clusteru souboru (horních 16 bitů)
`wrtTime`	22	2	čas posledního zápisu do souboru
`wrtDate`	24	2	datum posledního zápisu do souboru
`fstClustLO`	26	2	číslo prvního clusteru souboru (spodních 16 bitů)
`fileSize`	28	4	velikost souboru (v bytech)

Tabulka 2: Vybrané informace uložené v bootsectoru

[7] `https://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc`

Názvy souboru jsou vždy ve tvaru 8.3, tj. 8 znaků jméno, 3 znaky přípona. Pokud je jméno nebo přípona souboru kratší, je volné místo doplněno mezarami (znak `0x20`), tečka není uložena, nerozlišují se malá a velká písmena a jméno souboru musí obsahovat minimálně jeden znak jiný než mezera. Pokud je první byte názvu souboru:

- znak `0xe5`, jedná se o neplatný záznam (např. smazaný soubor).
- znak `0x00`, jedná se o neplatný záznam a příznak, že další zaznamy v adresáři jsou taky neplatné. (Není tak nutné procházet další záznamy.)

Atributy souboru jsou následující:

```
ATTR_READ_ONLY    0x01
ATTR_HIDDEN       0x02
ATTR_SYSTEM       0x04
ATTR_VOLUME_ID    0x08
ATTR_DIRECTORY    0x10
ATTR_ARCHIVE      0x20
ATTR_LONG_NAME    ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID
```

1.3 FAT tabulka

V adresáři máme u každého souboru uloženu informaci, kde se nachází první cluster souboru. Abychom mohli najít další clustery souboru, obsahuje souborový systém tabulku FAT, která pro daný cluster souboru ukazuje na následující cluster, pokud existuje. Pokud takový cluster neexistuje, je tam uložen příznak EOC (end of chain), což je hodnota větší nebo rovna `0x0FFFFFF8`. Z pohledu implementace můžeme na tabulku FAT nahlížet jako na pole 32bitových hodnot.

Pozor: První datový cluster je cluster s číslem 2.

2 Implementace

Známe-li strukturu souborového systému, vytvoření kódu, který s tímto souborovým systém pracuje, je přímočaré. Jelikož naše implementace bude postavena čistě na kódu v uživatelském prostoru, potřebujeme nějakým způsobem emulovat pevný disk. Zde se nabízí použít běžný soubor, se kterým budeme pracovat jako s blokovým zařízením, tj. budeme číst (popř. zapisovat) vždy celé sektory, např. 512 B.

2.1 Strukturované datové typy

Informace, které jsme si popsali v úvodní kapitole, budeme potřebovat uložit do vhodných datových typů. Pro reprezentaci souborového systému zavedeme typ `struct fat32`, který kromě výše popsaných informací obsahuje odkaz na soubor (image), kde je daný souborový systém uložen a další praktické informace, jmenovitě:

- první datový sektor (`FirstDataSector`),

- počet datových sektorů (`DataSec`),
- počet datových clusterů (`CountOfClusters`),
- velikost cluster (`ClusterSz`).

Kód této struktury vypadá následovně a najdeme jej v souboru `fat32.h`.

```
struct fat32 {
        FILE    *image;
        // data ulozena na disku
        uint16_t BPB_BytsPerSec;
        uint8_t  BPB_SecPerClus;
        uint16_t BPB_RsvdSecCnt;
        uint8_t  BPB_NumFATs;
        uint32_t BPB_TotSec32;
        uint32_t BPB_FATSz32;
        uint16_t BPB_ExtFlags;
        uint16_t BPB_FSVer;
        uint32_t BPB_RootClus;
        uint16_t BPB_FSInfo;
        uint8_t  BS_BootSig;
        uint32_t BS_VolID;
        char     BS_VolLab[11];
        char     BS_FilSysType[8];
        // data odvozena
        uint32_t ClusterSz;
        uint32_t FirstDataSector;
        uint32_t DataSec;
        uint32_t CountOfClusters;
};
```

V podobném duchu si zavedeme strukturovaný datový typ `struct fat32_dir_entry` (viz následující kód) představující záznamy v adresářích. Abychom si práci zjednodušili, informace o čase budeme ignorovat. A dále si zavedeme atribut, který sloučí do jedné hodnoty pozici prvního clusteru v souboru.

```
struct fat32_dir_entry {
        // data ulozena na disku
        char     fileName[11];
        uint8_t  attr;
        uint16_t fstClusterHI;
        uint16_t fstClusterLO;
        uint32_t fileSize;
        // data odvozena
```

```
        uint32_t fstClusterId;
};
```

Vedle těchto dvou datových typů si zavedeme ještě strukturovaný datový typ `struct fat32_fd`, který bude reprezentovat otevřený soubor (resp. adresář) a který má následující atributy.

```
struct fat32_fd {
    struct fat32 *fs;        // souborovy system
    uint32_t fileOffset;     // aktualni pozice v souboru
    uint32_t fileSize;       // velikost souboru
    uint32_t currentCluster; // aktualni cluster
    uint32_t bufOffset;      // aktualni pozice v bufferu
    uint32_t bufSize;        // pocet platnych bytu v bufferu
    int      isDirectory;
    uint8_t  buf[];          // buffer s nactenymi daty
};
```

2.2 Pomocné funkce

Nejdříve si zavedeme pomocnou funkci, která nám umožní pracovat se souborem jako s blokovým zařízením. Funkce `sector_read` slouží k tomu, abychom ze souborového systému `fs` načetli sektor `sector` a uložili jej do bufferu `buf`. Aby tato funkce byla obecnější, obsahuje ještě parametr `count`, který udává, kolik sektorů za sebou se má načíst.

```
static void sector_read(struct fat32 *fs, uint32_t sector, uint32_t count, uint8_t *buf) {
    fseek(fs->image, (long) sector * fs->BPB_BytsPerSec, SEEK_SET);
    fread(buf, fs->BPB_BytsPerSec, count, fs->image);
}
```

Tuto pomocnou funkci využijeme k vytvoření další pomocné funkce a to jest funkce, která přečte zadaný cluster:

```
static void cluster_read(struct fat32 *fs, uint32_t clusterId, uint8_t *buf) {
    uint32_t sector = ((clusterId - 2) * fs->BPB_SecPerClus) + fs->FirstDataSector;
    sector_read(fs, sector, fs->BPB_SecPerClus, buf);
}
```

2.3 Inicializace souborového systému

Než můžeme začít pracovat se soubory v souborovém systému, musíme provést inicializaci odpovídajících datových struktur. Postupujeme tak, že přečteme první sektor a z boot sectoru přečteme jednotlivé hodnoty, např. následovně.

```
void fat32_init(struct fat32 *fs, FILE *image) {
    uint8_t sector[512];
```

```
    fread(sector, 1, 512, image);
    fs->image = image;
    memcpy(&fs->BPB_BytsPerSec, sector + 11, 2);
    memcpy(&fs->BPB_SecPerClus, sector + 13, 1);
    // ...
```

Poznámka: V tomto případě jsme si (pro přehlednost) kód zjednodušili. Vícebytové hodnoty jsou v případě souborového systému uloženy ve formátu little-endian, což je stejný formát, jaký používají procesory x86, není proto potřeba konverze a můžeme použít funkci `memcpy`. Pokud bychom chtěli náš kód použít na procesorech pracujících s hodnotami big-endian, museli bychom provést odpovídající konverzi.

2.4 Otevření a čtení souboru

Pro otevření souboru si vytvoříme funkci `fat32_open`, která inicializuje strukturu `struct fat32_fd`, tj.

1. alokuje tuto strukturu,
2. alokuje prostor pro buffer, který má velikost jednoho clusteru,
3. nastaví výchozí hodnoty,
4. načte do bufferu první cluster souboru, pokud existuje.

Při čtení obsahu souboru (viz funkce `fat32_read` v souboru `fat32.c`) postupujeme tak, že čteme data z bufferu, který je součástí `struct fat32_fd`, a nejsou-li v tomto bufferu další data k dispozici, načteme do něj obsah dalšího clusteru, viz funkce `fat32_next_cluster`. K zjištění, kde se nachází další cluster souboru, slouží funkce `fat32_next_cluster_id`:

```
1 static uint32_t fat32_next_cluster_id(struct fat32 *fs, uint32_t currentCluster) {
2     uint8_t  buf[fs->BPB_BytsPerSec];
3     uint32_t fat_offset = currentCluster * 4;
4     uint32_t fat_sector = fs->BPB_RsvdSecCnt + (fat_offset / fs->BPB_BytsPerSec);
5     uint32_t fat_entry  = fat_offset % fs->BPB_BytsPerSec;
6     sector_read(fs, fat_sector, 1, buf);
7     uint32_t rawId = *((uint32_t *) (buf + fat_entry));
8     return rawId & 0x0fffffff;
9 }
```

Tato funkce přímo pracuje s tabulkou FAT. Připomeňme, že se jedná o pole (uložené na disku), kde každá jedna (32bitová) položka obsahuje číslo následujícího clusteru.

Nejdříve určíme offset položky (v bytech), která nás zajímá (hodnota `fat_offset`). Z něj odvodíme sektor, kde se tento záznam nachází (hodnota `fat_sector`) a odpovídající byte v rámci tohoto sektoru (hodnota `fat_entry`).

Nyní můžeme sektor přečíst (řádek 6) a přečíst i odpovídající záznam (řádek 7) a to tak, že přetypujeme pole bytů na pole 32bitových hodnot. Protože podle specifikace cluster může být jen 28bitová hodnota, ještě provedeme zúžení na 28bitů (řádek 8).

2.5 Čtení obsahu adresáře

Při čtení obsahu adresáře postupujeme analogicky jako v případě čtení obsahu souboru, tj. postupně čteme 32bytové položky z bufferu, který je součástí struktury `struct fat32_fd`, viz funkce `fat32_read_dir`. Pokud tato funkce narazí na platný záznam, vrátí kladnou hodnotu a naplní zadanou strukturu `struct fat32_dir_entry`. Pokud v adresáři nejsou žádné další platné záznamy, vrátí funkce hodnotu `0`. Všimněme si, že tato funkce funguje de facto jako iterátor nad obsahem adresáře.

3 Ukázková aplikace

Součástí přiložených zdrojových kódů je i aplikace umožňující zobrazit metadata souborového systému i jednotlivých souborů, a taky zobrazit obsah souboru. Pro vyzkoušení je přiložen i souborový systém, který byl vytvořený pomocí příkazů:

```
dd if=/dev/zero of=tutorial09.fat32 bs=4096 count=32768
/sbin/mkfs.fat -F 32 -s 2 tutorial09.fat32
```

Úkol č. 1: Vyzkoušejte si, jak se změní parametry souborového systému, pokud zvolíme jinou velikost (viz příkaz `dd`) nebo jiný počet sektorů na cluster (přepínač `-s`).

Operační systém Linux umožňuje připojit takto vytvořený souborový systém jako běžný adresář pomocí příkazu:

```
mount -o loop tutorial09.fat32 /mnt/foo
```

Provedení tohoto příkazu však vyžaduje administrátorská oprávnění.

Úkol č. 2: Vytvořte si nástroj, který vám umožní přečíst všechny soubory (i ve vnořených adresářích) v souborovém systému FAT32 a uložit je na (jiný) disk.

Úkol č. 3: Použijte tento nástroj.