Sisteme de FIȘIERE

Ciprian Ileană, Claudiu Soroiu

În cadrul acestui articol vom continua prezentarea sistemului de fișiere folosit de sistemele de operare produse de Microsoft. În articolul din numărul anterior am prezentat câteva structuri de date importante prezente pe un volum FAT. Vom descrie acum modul în care pot fi inițializate volumele FAT, structura unor sectoare specifice sistemului FAT32, structura directoarelor, intrările cu nume lungi din FAT etc.

Inițializarea volumelor FAT

Sistemele de operare *Microsoft* folosesc doar sistemul *FAT12* pentru dischete. Deoarece numărul formatelor de dischete care au aceeași dimesiune este limitat, inițializarea *BPB*-ului se realizează cu ajutorul unui tabel simplu.

Nu există nici o metodă dinamică de calcul pentru *FAT12*. Astfel, pentru formatele *FAT12* toate calculele pentru `BPB_SecPerClus` și `BPB_FATSz16` au fost realizate folosindu-se foaia și creionul, iar rezultatele au fost înregistrate în tabele. Una dintre condițiile impuse a fost, evident, ca numărul de *cluster*-e rezultat să fie mai mic decât 4085. În cazul în care discul (partiția) folosit(ă) are o capacitate mai mare de 4 MB folosirea sistemului *FAT12*, nu mai este recomandată. Este indicată folosirea unei valori mai mici pentru `BPB_SecPerClus`, astfel încât volumul să devină *FAT16*.

Din acest punct și până la sfârșitul acestei secțiuni ne vom referi doar la discuri (partiții) care au 512 bytes pe sector. Trebuie avut în vedere faptul că informațiile din această secțiune nu le veți putea folosi decât în cazul discurilor (partițiilor) care au 512 bytes pe sector.

Există o valoare fixată care face separarea între *FAT16* și *FAT32*. Orice volum cu o dimensiune mai mică decât aceasta este *FAT16* și orice volum de această dimensiune, sau mai mare, este *FAT32*. De exemplu, pentru *Windows* această valoare este de 512 MB. Orice volum *FAT* mai mic de 512 MB este *FAT16* și orice volum *FAT* de 512 MB, sau mai mare, este *FAT32*. Totuși, există multe volume *FAT16* care sunt mai mari de 512 MB, datorită variatelor modalități prin care se poate forța utilizarea unui *FAT16* în locul prestabilitului *FAT32* prin modificarea limitelor în care acestea se încadrează.

Acum vom vorbi despre limita prestabilită pentru volumele *MS-DOS* și *Windows* care nu au fost încă formatate. Există două tabele, unul pentru *FAT16* iar celălalt pentru *FAT32*.

```
struct DskSzToSecPerClus{
  DWORD DiskSize;
  BYTE SecPerClusVal;
};

/* Tabela pentru volume FAT16. */
DskSzToSecPerClus DskTableFAT16[]={
  {8400,0}, // discuri cu capacitate de
            // pana la 4.1 MB, valoarea
            // 0 pentru SecPerClusVal
            // reprezinta o eroare
  {32680,2}, // discuri cu capacitate de
             // pana la 16 MB,cluster de
             // 1k
  {262144,4}, // discuri cu capacitate
              // de pana la 128 MB,
              // cluster de 2k
  {524288,8}, // discuri cu capacitate
              // de pana la 256 MB,
              // cluster de 4k
  {1048576,16}, // discuri cu capacitate
                // de pana la 512 MB,
                // cluster de 8k
// Intrarile care urmeaza nu sunt folosite
// decat in cazul in care FAT16 este fortat
  {2097152,32}, // discuri cu capacitate
                // de pana la 1 GB,
                // cluster de 16k
  {4194304,64}, // discuri cu capacitate
                // de pana la 2 GB,
                // cluster de 32k
```

```
  {0xFFFFFFFF,0} // orice disc cu o
                 // capacitate mai mare
                 // de 2GB, valoarea 0
                 // pentru SecPerClusVal
                 // reprezinta o eroare
};

/* Tabela pentru volumele FAT32. */
DSKSzToSecPerClus DskTableFAT32[]={
  {66600,0}, // discuri cu capacitate de
             // pana la 32.5 MB,
             // valoarea 0 pentru
             // SecPerClusVal reprezinta
             // o eroare
  {532480,1}, // discuri cu capacitate
              // de pana la 260 MB,
              // cluster de 5k
  {16777216,8}, // discuri cu capacitate
                // de pana la 8 GB,
                // cluster de 4k
  {33554432,16}, // discuri cu capacitate
                 // de pana la 16 GB,
                 // cluster de 8k */
  {67108864,32}, // discuri cu capacitate
                 // de pana la 32 GB,
                 // cluster de 16k
  {0xFFFFFFFF,64} // discuri cu capaci-
                  // tate mai mare de
                  // 32GB, cluster de 32k
};
```

Așadar, fiind dată dimensiunea unui disc și un tip de *FAT* (*FAT16* sau *FAT32*), avem acum o valoare pentru `BPB_SecPerCluster`. Singura operație care mai trebuie efectuată este calcularea numărului de sectoare ocupate de structura *FAT* pentru a putea seta valoarea `BPB_FATSz16` sau `BPB_FATSz32`.

În cele ce urmează vom presupune că `BPB_RootEntCnt`, `BPB_RsvdSecCnt` și `BPB_NumFATs` sunt setate corect. De asemenea, presupunem că variabila `DskSize` reprezintă mărimea volumului pe care o vom atribui variabilei `BPB_TotSec32` sau variabilei `BPB_TotSec16`.

```
RootDirSectors=((BPB_RootEntCnt*32)+
      (BPB_BytsPerSec-1))/BPB_BytsPerSec;
TmpVal1=DskSize-(BPB_ResvdSecCnt+
                         RootDirSectors);
TmpVal2=(256*BPB_SecPerClus)+BPB_NumFATs;
if (FATType==FAT32)
  TmpVal2=TmpVal2/2;
FATSz=(TMPVal1+(TmpVal2-1))/TmpVal2;
if (FATType==FAT32){
  BPB_FATSz16=0;
  BPB_FATSz32=FATSz;
}
else
  BPB_FATSz16 = LOWORD(FATSz);
  // nu exista BPB_FATSz32 intr-un BPB de
  // FAT16
```

Matematica ce stă la baza calculelor din codul de mai sus este foarte complicată. Important este faptul că sistemele de operare *Microsoft* funcționează folosindu-se de aceste calcule. Totuși, aceste calcule matematice nu funcționează perfect. Pot apărea erori cum ar fi setarea valorii

Structura sectorului FSInfo

Tabelul 1

Denumire	*Offset*	*Dimensiune*	*Descriere*
`FSI_LeadSig`	0	4 bytes	Are valoarea `0x41615252`. Această semnătură este folosită pentru a valida faptul că acest sector este un sector `FSInfo`.
`FSI_Reserved1`	4	480 bytes	Acest câmp este în prezent rezervat pentru dezvoltări ulterioare; codul *FAT32* ar trebui să inițializeze acești bytes cu `0` și nu ar trebui să fie folosiți în prezent.
`FSI_StrucSig`	484	4 bytes	Are valoarea `0x61417272`; reprezintă o altă semnătură care este adesea folosită în sector, în apropierea câmpurilor folosite.
`FSI_Free_Count`	488	4 bytes	Conține ultimul *cluster* liber cunoscut; dacă valoarea sa este `0xFFFFFFFF` înseamnă că ultimul *cluster* liber nu este cunoscut și trebuie determinat folosind alte metode.
`FSI_Reserved2`	496	12 bytes	Acest câmp este în prezent rezervat pentru dezvoltări ulterioare; codul *FAT32* ar trebui să inițializeze acești bytes cu `0` și nu ar trebui să fie folosiți în prezent.
`FSI_TrailSig`	508	4 bytes	Are valoarea `0xAA550000`; această semnătură este folosită pentru a valida faptul că acest sector este un sector `FSInfo`. Trebuie remarcat faptul că valorile octeților cu *offset*-urile 510 și 511 sunt identice cu valorile octeților corespunzători din sectorul 0.

FATSz cu până la două sectoare în plus în cazul unui *FAT16* și cu până la opt sectoare în plus în cazul unui *FAT32*.

Dacă în urma calculelor valoarea rezultată pentru FATSz va fi mai mică decât cea reală, aceasta va determina funcționarea defectoasă a *FAT*-ului. În concluzie, este preferată metoda de calcul de mai sus, cu toate că în unele cazuri se vor irosi câteva sectoare (este cunoscut faptul că *FAT*-ul funcționează corect și în cazul unei valori mai mari decât cea reală pentru FATSz).

Sectoarele FSInfo și BkBoot

În cazul unui volum *FAT32*, structura *FAT* nu are o limită de mărime propriu-zisă așa cum este în cazul *FAT16* și *FAT12*. Din acest motiv, este alocat un spațiu destinat stocării ultimului *cluster* liber cunoscut din cadrul volumului *FAT32*, optimizând astfel lucrul aplicațiilor care în loc să calculeze mereu valoarea acestui *cluster* vor avea acum acces direct la aceasta. Numărul sectorului FSInfo se regăsește în valoarea câmpului BPB_FSInfo; în cazul sistemelor de operare *Microsoft* această valoare este setată întotdeauna cu 1. Structura sectorului FSInfo este prezentată în tabelul 1.

O altă facilitate existentă în *FAT32*, dar nu și în *FAT16* sau *FAT12*, este prezența câmpului BPB_BkBootSec. Cu toții știm că volumele *FAT16/FAT12* pot fi în totalitate pierdute în cazul în care conținutul sectorului 0 este rescris sau este distrus nemaiputând fi citit. Prezența câmpului BPB_BkBootSec (are valoarea 6 întotdeauna) în cazul volumelor *FAT32* reduce șansele de pierdere a informațiilor conținute în respectivul volum, deoarece pe al BPB_BkBootSec-ulea sector se află o copie la zi a sectorului de *boot*, copie ce conține și *BPB*-ul volumului.

În cazul în care informațiile din sectorul 0 au fost rescrise din greșeală, utilitarele de disc au simpla sarcină de a restaura datele originale folosindu-se de copia prezentă în sectorul de *back-up*. Dacă sectorul 0 devine inutilizabil și prin urmare nu mai poate fi citit, prezența sectorului de *back-up* permite folosirea volumului în continuare, însă este recomandată salvarea informației și schimbarea discului. Acest al doilea caz (sectorul 0 devine "bad") este motivul pentru care câmpul BPB_BkBootSec nu ar trebui să conțină altă valoare decât 6. Mai exact, multe din sistemele de operare caută acest sector de *back-up* începând cu al șaselea sector al volumului *FAT32*.

Acest așa-zis sector de *back-up* ocupă, în realitate, trei sectoare, fiecare având o lungime de 512 bytes. De asemenea, trebuie remarcat faptul că, la fel ca și sectorul de *boot*, cele trei sectoare conțin la *offset*-urile 510 și 511 semnătura 0xAA55.

Structura directoarelor FAT

Pentru început vom ignora intrările de directoare cu nume lungi și ne vom ocupa doar de intrările de directoare cu nume scurte. Un director din *FAT* nu este altceva decât un "fișier" compus dintr-o listă liniară de structuri pe 32 de bytes. În cazul volumelor *FAT12* și *FAT16* directorul rădăcină are o locație fixă pe disc (imediat după ultimul *FAT*) și o dimensiune fixă calculată în sectoare (vezi calculele pentru RootDirSectors din numărul anterior al revistei ***GInfo***); în acest caz primul sector al directorului rădăcină este relativ la primul sector al volumului *FAT*:

```
FirstRootDirSecNum BPB_ResvdSecCnt+
      (BPB_NumFATs*BPB_FATSz16);
```

În cazul sistemului *FAT32*, directorul rădăcină poate avea dimensiune variabilă și este o listă de *cluster*-e, la fel ca orice alt director. Primul *cluster* al directorului rădăcină pe volumele *FAT32* este stocat în BPB_RootClus. Spre deosebire de celelalte directoare, directorul rădăcină nu are setate o dată și o oră, nu poate avea alt nume în afară de "\" și nu conține fișierele "." și ".." ca fiind primele două intrări în director. O mare deosebire față de directoarele obișnuite este aceea că directorul rădăcină este singurul director din cadrul volumului *FAT* care poate avea setat doar bitul corespunzător atributului ATTR_VOLUME_ID.

Structura *FAT* a unui director este prezentată în tabelul 2.

DIR_Name

Dacă DIR_Name[0]=0xE5, atunci intrarea directorului este liberă (nu există nici un nume de fișier sau de director în această intrare).

Dacă DIR_Name[0]=0x00, atunci intrarea directorului este liberă (la fel ca în cazul 0xE5) și după această intrare în director nu mai este alocată nici o altă intrare de director (toți octeții DIR_Name[0], din toate intrările aflate după aceasta, sunt setați cu valoarea 0). Valoarea 0, spre deosebire de 0xE5, indică *driver*-ului sistemului *FAT*, faptul că celelalte intrări de directoare nu mai trebuie examinate deoarece se știe deja că sunt libere.

Dacă Dir_Name[0]=0x05, atunci acest byte are, de fapt, valoarea 0xE5. Valoarea 0x05 este folosită pentru a evita confundarea caracterului cu marcajul unei intrări libere de director.

Valorile nefolosite pentru numele directorului sunt completate cu spații (caractere cu codul 0x20). Totuși, DIR_Name[0] nu poate avea valoarea 0x20.

Între numele și extensia unui director apare caracterul implicit '.'. Acest caracter nu este stocat în DIR_name.

Câmpul DIR_name nu poate conține nici unul dintre caracterele:

- literele mici ale alfabetului;
- valori cuprinse între 0x01 și 0x19, cu excepția valorii 0x05;
- valorile 0x22, 0x2A, 0x2B, 0x2C, 0x2E, 0x2F, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x5B 0x5C, 0x5D și 0x7C.

În *FAT* numele tuturor directoarelor sunt unice.

DIR_Attr

Câmpul DIR_Attr este folosit pentru a specifica atributele unui fișier.

Semnificațiile biților sunt următoarele:

Intrarea pentru un director

Tabelul 2

Denumire	*Offset*	*Dimensiune*	*Descriere*
DIR_Name	0	11 bytes	Nume scurt.
DIR_Attr	11	1 byte	Atributele fișierului: ATTR_READ_ONLY - bitul 0 ATTR_HIDDEN - bitul 1 ATTR_SYSTEM - bitul 2 ATTR_VOLUM_ID - bitul 3 ATTR_DIRECTORY - bitul 4 ATTR_ARCHIVE - bitul 5 Cei mai semnificativi doi biți din byte-ul atributelor sunt rezervați și ar trebui întotdeauna setați pe 0 la crearea fișierului fără a fi ulterior modificați.
DIR_NTRes	12	1 byte	Rezervat pentru a fi folosit de *Windows NT*; valoarea este setată cu 0 la crearea fișierului fără a fi ulterior modificată.
DIR_CrtTimeTenth	13	1 byte	Milisecundele din momentul creării fișierului. În realitate, sunt numărate doar zecimile de secundă.
DIR_CrtTime	14	2 bytes	Ora la care a fost creat fișierul.
DIR_CrtDate	16	2 bytes	Data la care a fost creat fișierul.
DIR_LstAccDate	18	2 bytes	Data ultimei accesări (citire/scriere).
DIR_FstClusHI	20	2 bytes	Cei mai semnificativi 16 biți ai numărului primului *cluster* al intrării (pentru *FAT12* sau *FAT16* are întotdeauna valoarea 0).
DIR_WrtTime	22	2 bytes	Ora ultimei modificări (creare/scriere).
DIR_WrtDate	24	2 bytes	Data ultimei modificări (creare/scriere).
DIR_FstClusLO	26	2 bytes	Cei mai semnificativi 16 biți ai numărului primului *cluster* al intrării.
DIR_FileSize	28	4 bytes	Dimensiunea fișierului pe 32 de biți (DWORD), exprimată în bytes.

- ATTR_READ_ONLY - indică faptul că scrierea asupra fișierului ar trebui să eșueze.
- ATTR_HIDDEN - indică faptul că la o listare obișnuită a fișierelor, acesta nu ar trebui afișat.
- ATTR_SYSTEM - indică faptul că acesta este un fișier al sistemului de operare.
- ATTR_VOLUME_ID - ar trebui să existe un singur astfel de "fișier" pe fiecare volum, iar acesta ar trebui să fie directorul rădăcină; numele acestui fișier este, de fapt, eticheta volumului. DIR_FstClusHI și DIR_FstClusLO trebuie să aibă întotdeauna valoarea 0 pentru etichetele de volum (pentru fișierele de etichetă nu se alocă nici un *cluster* de date).
- ATTR_DIRECTORY - indică faptul că acest fișier este un director (*container* de fișiere) care conține intrările altor fișiere.
- ATTR_ARCHIVE - acest atribut vine în ajutorul utilitarelor de *back-up*; *driver*-ul sistemului de fișiere *FAT* setează acest bit în cazul în care fișierul a fost creat, redenumit sau modificat; astfel utilitarele de *back-up* pot folosi acest atribut pentru a indica fișierele din volum care au fost modificate de la ultimul *back-up* efectuat.

În cazul în care atributul ATTR_LONG_NAME este setat acesta indică faptul că "fișierul" este parte a unui nume lung al unui alt fișier.

Când un director este creat (un fișier care are setat atributul ATTR_DIRECTORY din câmpul DIR_Attr), câmpul DIR_FileSize este setat cu 0 deoarece directoarele sunt dimensionate prin simpla urmărire a listei de *cluster*-e până la întâlnirea marcajului *EOC*. La creare, directorului i se alocă un *cluster* (acest lucru nu este valabil și în cazul sistemelor *FAT16*/*FAT12*) al cărui număr va fi scris în DIR_FstClusLO și DIR_FstClusHi; după aceasta la intrarea acestui *cluster* în *FAT* se va scrie marcajul *EOC*. În cazul în care directorul creat este directorul rădăcină, nu mai trebuie efectuate alte operații.

Dacă se creează un director obișnuit, vor trebui create două intrări speciale în primii 32 de bytes ai intrării directorului (primii 32 de bytes din zona de date a *cluster*-ului care tocmai a fost alocat).

Aceste două intrări vor avea valoarea DIR_Name setată astfel:

- pentru prima avem DIR_Name = ".   ".
- pentru a doua avem DIR_Name = "..   ".

Aceste două intrări poartă numele de intrări *dot* (punct) și *dotdot* (punct punct). Ceea ce este important de reținut la aceste intrări este faptul că intrarea *dot* este un director care se referă la el însuși, iar intrarea *dotdot* referă *cluster*-ul directorului părinte (acesta este 0 în cazul în care directorul părinte este directorul rădăcină).

Formatele de dată și timp

Multe sisteme de fișiere *FAT* nu suportă decât câmpurile `DIR_WrtTime` și `DIR_WrtDate`. Din acest motiv, câmpurile `DIR_CrtTimeMil`, `DIR_CrtTime`, `DIR_CrtDate` și `DIR_LstAccDate` sunt opționale (este obligatorie tratarea câmpurilor `DIR_WrtTime` și `DIR_WrtDate`). În cazul în care aceste câmpuri nu sunt tratate, acestea ar trebui să fie inițializate cu valoarea 0 și ignorate de către celelalte operații cu fișiere.

Data este stocată într-o intrare *FAT* pe16 biți și este relativă la "*era MS-DOS*" (prima dată este 01/01/1980). Formatul datei este prezentat mai jos (bitul 0 este *LSB* iar bitul 15 este *MSB*):

- biții `0-4`: ziua lunii, cu valori valide între `1` și `31` (inclusiv).
- biții `5-8`: luna anului, cu valori valide între `1` și `12` (inclusiv).
- biții `9-15`: numărul de ani scurși din `1980` și până în prezent, cu valori valide între `0` și `127` (inclusiv); se acoperă perioada `1980-2107`.

Timpul este stocat într-o intrare *FAT* pe 16 biți cu o granulație de două secunde. Formatul timpului este prezentat mai jos (bitul 0 este *LSB* iar bitul 15 este *MSB*):

- biții `0-4`: interval de două secunde, cu valori valide între `0` și `29` (inclusiv); se acoperă intervalul `0-58` secunde).
- biții `5-10`: minutele, cu valori valide între `0` și `59` (inclusiv).
- biții `11-15`: ora, cu valori valide între `0` și `23` (inclusiv).

Se acoperă astfel un interval cuprins între `00:00:00` și `23:59:58` (acesta este intervalul valorilor valide ale timpului).

Intrări cu nume lungi

Adăugarea intrărilor cu nume lung în *FAT* a venit într-un moment crucial pentru design-ul existent al *FAT*-ului:

- Modificările trebuiau să fie transparente pentru versiunile anterioare ale sistemului de operare *MS-DOS*. Scopul principal este acela, ca aplicațiile realizate pentru versiunile anterioare de *MS-DOS/Windows* să nu poată accesa cu ușurință intrările cu nume lung. Singurele aplicații *MS-DOS* care pot "să vadă" intrările cu nume lung sunt aplicațiile bazate pe căutare utilizând ***FCB***-ul (***F**ile **C**ontrol **B**lock*), acest lucru se întâmplă numai în cazul în care folosesc masca completă de căutare cu caractere de control (`*.*`) și atributul complet de căutare cu toți biții setați (`0xFF`). În versiunile de *MS-DOS/Windows* anterioare sistemului de operare *Windows '95*, nici o aplicație *MS-DOS* nu putea "să găsească" nici măcar o singură intrare cu nume lung.
- O intrare cu nume lung trebuie să fie alocată fizic, pe disc, în apropierea intrării cu nume scurt la care aceasta face referire. După cum se va vedea mai târziu, intrările cu nume lung se află imediat înaintea intrării cu nume

Intrările cu nume lung

Tabelul 3

Denumire	*Offset*	*Dimensiune*	*Descriere*
`LDIR_Ord`	`0`	`1` byte	Numărul de ordine al intrării în setul de intrări pentru numele lung atașat unui nume scurt.
`LDIR_Name1`	`1`	`10` bytes	Primele cinci caractere din subcomponenta numelui lung atașat unei intrări de director; se folosește setul de caractere *UNICODE*.
`LDIR_Attr`	`11`	`1` byte	Atributele intrării - bitul `ATTR_LONG_NAME` trebuie să aibă valoarea 1.
`LDIR_Type`	`12`	`1` byte	Dacă are valoarea `0`, atunci intrarea este o sub-componentă a unui nume lung. Alte valori sunt rezervate pentru versiunile următoare. O valoare diferită de `0` indică o intrare de alt tip.
`LDIR_Chksum`	`13`	`1` byte	Suma de control a numelui din intrarea de director cu nume scurt.
`LDIR_Name2`	`14`	`12` bytes	Caracterele `6-11` din subcomponenta numelui lung din această intrare a directorului.
`LDIR_FstClusLO`	`26`	`2` bytes	Trebuie să aibă valoarea `0` pentru compatibilitatea cu versiunile precedente ale sistemelor de operare *Microsoft*.
`LDIR_Name2`	`28`	`4` bytes	Ultimele două caractere din subcomponenta numelui lung din această intrare a directorului.

scurt cu care sunt asociate, și existența lor are un impact insesizabil asupra performanței sistemului de fișiere.
- Dacă un utilitar de disc detectează intrări cu nume lung, acesta nu pune în pericol integritatea datelor existente în fișiere. Utilitarele de disc, de obicei, nu utilizează aplicațiile *MS-DOS* ca să acceseze structurile de date specifice sistemului de fișiere de pe disc. Acestea citesc informația din sectorul fizic sau logic de pe disc și verifică ce conține intrarea din director. Intrările cu nume lung au fost adăugate sistemului de fișiere *FAT* astfel încât să nu cauzeze pierderi de date pe un disc care conține aceste tip de intrări dacă discul a fost "reparat" de un utilitar compatibil doar cu versiunile de *MS-DOS/Windows* anterioare sistemului de operare *Windows '95*.

Din dorința celor de la *Microsoft*, ca numele lungi să se impună pe aceleași structuri *FAT* care au fost construite începând cu prima versiune *MS-DOS*, intrările cu nume lung sunt definite ca o intrare cu nume scurt, dar cu atribute speciale. O intrare cu nume lung este o intrare obișnuită cu nume scurt în care câmpul atribut are valoarea:

```
ATTR_LONG_NAME=ATTR_READ_ONLY|
ATTR_HIDDEN|ATTR_SYSTEM|ATTR_VOLUME_ID;
```

Pentru a determina dacă o intrare este o componentă a unui nume lung, masca atributului este următoarea:

```
ATTR_LONG_NAME=ATTR_READ_ONLY|
ATTR_HIDDEN|ATTR_SYSTEM|ATTR_VOLUME_ID|
ATTR_DIRECTORY|ATTR_ARCHIVE;
```

Când este întâlnită o astfel de intrare, aceasta este tratată separat de sistemul de fișiere (operare). Aceste intrări sunt tratate ca o parte a unui set de intrări din director care sunt asociate unei singure intrări cu nume scurt din director. Fiecare intrare de nume lung are structura din tabelul 3.

Organizarea și asocierea dintre intrările cu nume lung și cele cu nume scurt

Fiecărei intrări cu nume scurt îi precede un set de intrări cu nume lung. Intrările cu nume lung sunt asociate intrării cu nume scurt dintr-un motiv foarte simplu: numai intrările cu nume scurt sunt vizibile sistemelor de operare anterioare sistemului *Windows '95*. Fără intrarea cu nume scurt asociată, intrările cu nume lung sunt complet invizibile acestor sisteme de operare. O intrare cu nume lung nu poate fi interpretată dacă nu se găsește intrarea cu nume scurt căreia îi este asociată. Dacă există intrări cu nume lung cărora nu le este asociată nici o intrare validă cu nume scurt, aceste intrări poartă numele de "orfani".

În primul rând, fiecare componentă a unui set de intrări cu nume lung este numerotat în mod unic, și valoarea numărului, ultimei componente a setului este calculată prin operația `OR` (SAU logic) aplicată între acesta și un număr care indică faptul că acest component este ultimul din set. Câmpul `LDIR_Ord` este folosit pentru a realiza această determinare a ultimului component al setului. Prima componentă a setului are valoarea `1` și este păstrată în câmpul `LDIR_Ord`. A *n*-a componentă are valoarea `n OR LAST_LONG_ENTRY` pentru acest câmp. Acest câmp (`LDIR_Ord`) nu poate avea nici una dintre valorile `0x00` sau `0xE5`. Aceste două valori au fost utilizate de la început de sistemul de fișiere pentru a indica o intrare "liberă" în director, respectiv "ultima" intrare de director din *cluster*. Câmpul `LDIR_Ord` nu poate avea aceste valori, doarece nu se află în domeniul variabilei (valorile pe care le poate avea variabila `LDIR_Ord` sunt cuprinse între 1 și `n OR LAST_LONG_ENTRY`). În cazul în care `LDIR_Ord` depășește domeniul se consideră că intrarea este alterată și este tratată ca un ***orfan*** de către sistemul de fișiere.

În al doilea rând, se calculează o sumă de control pe 8 biți, pe baza numelui scurt din intrare cu nume scurt din director, în momentul în care sunt create intrările cu nume lung și scurt. Pentru a calcula această sumă de control se folosesc toate cele 11 caractere ale numelui din intrarea cu nume scurt. Suma de control este stocată în fiecare intrare cu nume lung atașată intrării cu nume scurt. Dacă una dintre sumele de control din setul de intrări cu nume lung nu corespunde cu suma de control a numelui conținut de intrarea cu nume scurt, atunci intrările cu nume lung sunt tratate ca orfani. Acest lucru se întâmplă atunci când intrările cu nume scurt sunt citite de o versiune mai veche de *MS-Dos/Windows* și la o redenumire a unei intrări care are atașat numele lung, se schimbă doar numele scurt.

Algoritmul de calculare a sumei de control este:

```
unsigned char ChkSum (unsigned char
                                    *pFcbName){
  short FcbNameLen;
  unsigned char Sum;
  Sum =0;
  for (FcbNameLen=11;FcbNameLen;
                                  FcbNameLen--)
    Sum=((Sum&1)?0x80:0)+(Sum>>1)+
                                   *pFcbName++;
  return Sum;
}
```

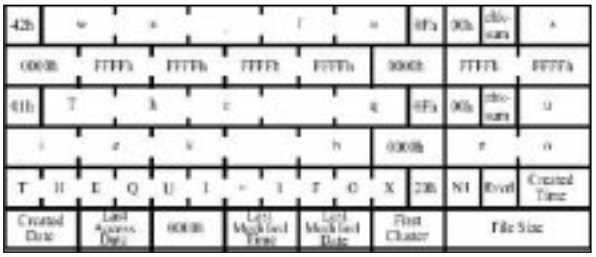
Intrarea cu nume scurt este o structură care conține câmpuri cum ar fi data ultimei accesări, ora creării, data creării, primul *cluster*, mărime etc. De asemenea, intrarea cu nume scurt conține un nume vizibil pentru versiunile mai vechi de *MS-DOS/Windows*. Intrările cu nume lung stochează informații noi și nu este nevoie să conțină informații deja existente în intrarea cu nume scurt. De fapt, intrările cu nume lung conțin numele lung al unui fișier. Numele conținut de intrarea cu nume scurt, căreia îi este asociat un set de intrări cu nume lung, poartă denumirea de ***alias al fișierului***.

Fișiere și subdirectoare cu nume lung

Un nume lung poate conține mai multe caractere decât încap într-o singură intrare din director. Când acest lucru se întâmplă, numele lung este stocat în mai multe intrări cu nume lung. În orice caz, câmpurile ce conțin numele, din interiorul unei intrări cu nume lung, sunt separate. Următorul exemplu ilustrează cum un nume lung este stocat în mai multe intrări din director. Numele lungi se termină cu

caracterul NULL (0x00) și completate cu 0xFFFF, pentru a evita coruperea câmpurilor unei intrări cu nume lung, de către utilitarele de disc existente deja. Dacă lungimea unui nume lung se potrivește exact în *n* intrări cu nume lung din director (un întreg multiplu de 13), atunci acesta nu se mai termină cu caracterul NULL și nu mai necesită completarea cu 0xFFFF.

Să presupunem că pe disc există un fișier cu numele: The quick brown.fox. Următorul exemplu ilustrează cum este împachetat numele în intrări cu nume lung și scurt. Majoritatea câmpurilor din intrările în directoare sunt și ele prezente.



Limita numelor și seturile de caractere

În această secțiune vă vom prezenta restricțiile care apar la numele fișierelor cu privire la numărul de caractere și seturile de caractere admise.

Intrări cu nume scurt

Numele scurte sunt limitate la 8 caractere urmate de un "." opțional și de o extensie de până la 3 caractere lungime. Lungimea totală a căii către fișier nu poate fi mai mare de 80 de caractere (64 de caractere pentru cale, 3 caractere pentru litera discului, 12 caractere pentru nume, punct și extensie și caracterul NULL). Caracterele pot fi litere, cifre sau caractere care au codul mai mare decât 127. Mai pot fi folosite și următoarele caractere speciale:

$ % ' - _ @ ~ ` ! ( ) { } ^ # &.

Numele sunt stocate în pagina de cod ***OEM*** (***O**riginal **E**quipment **M**anufacturer*) pentru care sistemul este configurat în momentul în care este creată o nouă intrare în director. Intrările cu nume scurt se pastrează în formatul *OEM*, pentru compatibilitate cu versiuni anterioare de *MS-DOS/Windows*. Caracterele *OEM* sunt caractere pe 8 biți sau pot fi bazate pe perechi de caractere ***DBCS*** (***D**ouble-**B**yte **C**haracter **S**et*) pentru anumite pagini de cod.

Numele scurte sunt acceptate de către sistemul de fișiere dacă se folosesc majusculele. O problemă care apare pentru aproape toate paginile de cod *OEM* este aceea că ele dețin informații despre conversia literelor mici în litere mari corespunzătoare, pentru caracterele extinse, dar o literă mare nu corespunde unei singure litere mici. Acest lucru se întâmplă din cauză că mai multe caractere extinse pot avea aceeași majusculă. Acest lucru creează probleme, deoarece nu se păstrează informația pe care o exprimă un singur caracter extins. Această legătură dintre literele mari și cele mici, are și câteva avantaje printre care acela că previne crearea unor fișiere cu nume care ar părea diferite, dar din cauza acestei legături, prin convertirea la litere mari se obține același nume de fișier.

Intrările cu nume lung

Numele lungi sunt limitate la 255 de caractere, fără caracterul NULL de la sfârșit. Lungimea totală a unui nume lung nu poate depăși 260 de caractere, inclusiv caracterul NULL. Caracterele admise sunt aceleași ca și în cazul numelor scurte, cu precizarea că "." poate fi folosit de mai multe ori într-un nume lung. Următoarele șase caractere pot fi folosite într-un nume lung, în plus față de cele permise la numele scurte: + , ; = [ ].

Spațiile dintr-un nume lung care se află la începutul și sfârșitul numelui, sunt ignorate. Sunt ignorate și caracterele "." care se află la sfârșitul numelui.

Numele lungi sunt stocate în intrările cu nume lung în format *UNICODE*. Nu este posibilă stocarea numelui scurt în format *UNICODE*, deoarece în acest caz caracterele sunt reprezentate pe 8 biți sau caractere *DBCS*.

Când sistemul de fișiere citește numele lung, nu mai face conversia la majuscule, deci numele lung se găsește în forma originală. *UNICODE*-ul rezolvă problema de conversie între literele mari și mici care apare în unele pagini de cod *OEM*, prin conversia la o singură literă mică, și invers, conversia la o majusculă unică.

Legătura dintre numele scurt și cel lung

Numele conținute în setul de intrări cu nume scurt în director poartă denumirea de ***spațiul numelui scurt***, iar numele conținute in setul de intrări cu nume lung poartă denumirea de ***spațiul numelui lung***. Împreună, ele formează un singur spațiu unificat în care nu există duplicate. Aceasta înseamnă că orice nume de fișier/director, fie el lung sau scurt, este unic (apare o singură dată în lista de fișiere și directoare). Mai mult, chiar dacă pentru numele lungi de fișiere/directoare este permisă și utilizarea literelor mici, nu pot exista două fișiere/directoare care să aibă același nume (după conversie). De exemplu, dacă dorim să creăm numele director, acesta nu poate fi creat dacă există deja pe disc, în directorul de lucru un fișier cu numele DIRECTOR sau cu numele lung Director.

Toate tipurile de operații ale sistemului de fișiere (căutare, deschidere, creare, ștergere, redenumire) sunt insensibile la literele mari și mici, deci dacă dorim să deschidem fișierul DIRECTOR, sistemul de fișiere va deschide automat fișierul care există, fie cu numele director sau Director. Aceleași reguli se aplică și în cazul caracterelor extinse.

O căutare pentru nume scurte de fișiere, verifică numai numele scurte dintr-un director, în timp ce o căutare pentru nume lungi de fișiere verifică și numele scurte și pe cele lungi. În timp ce sistemul de operare parcurge directorul, el determină numele lungi de fișiere conținute de sistemul de fișiere. Când se întâlnește un nume scurt, care este asociat numelui lung, operația de căutare pentru nume

lungi verifică mai întâi dacă numele lung și după aceea cel scurt se potrivesc cu masca de căutare.

Dacă un caracter de pe disc (stocat în format *OEM* sau *UNICODE*) nu poate fi tradus într-un caracter corespunzător paginii de cod folosite (*OEM* sau *ANSI*), el va fi înlocuit la returnarea numelui găsit cu caracterul `"_"`. Acest caracter este același în toate paginile de cod *OEM* și *ANSI*.

Convențiile de nume și numele lungi

O aplicație permite utilizatorului să specifice numele lung care să fie atribuit unui fișier sau unui director, dar nu-i permite să-și aleagă și numele scurt al acestuia. Aceasta se datorează faptului că spațiul numelor scurte și spațiul numelor lungi sunt considerate ca fiind un singur spațiu unificat. După cum se poate observa, nu este posibil să avem două fișiere cu același nume lung și cu nume scurte diferite. Această restricție a fost impusă pentru a preveni confuzia din rândul utilizatorilor, și aplicațiilor, în ceea ce privește numele propriu al fișierului sau directorului. Pentru transparența acestei restricții, atunci când avem un nume lung de fișier sau director, și dorim să-l creăm, aplicația verifică dacă există numele lung și dacă nu există, generează automat numele scurt atașat numelui lung, pe baza numelui lung, astfel încât să nu existe alt fișier/director care să aibă același nume scurt cu cel care va fi creat.

Tehnica aleasă pentru generarea automată a numelor scurte din nume lungi este preluată și adaptată din *Windows NT*. Numele scurt generat automat este compus din două părți: ***numele de bază*** și ***coada numerică***.

Efectul intrărilor cu nume lung în directoare asupra versiunilor precedente de FAT

Suportarea numelor lungi de fișiere este foarte importantă pentru discul hard. Numele lungi sunt suportate și de mediile de date portabile (*removable media*). Implementarea permite suportul pentru nume lungi de fișiere, fără să încalce compatibilitatea cu formatul existent de *FAT*. Un disc poate fi citit de un sistem mai vechi fără probleme de compatibilitate. Un disc deja creat, nu trebuie să fie transformat pentru a suporta numele lungi. Toate fișierele existente rămân neschimbate. Intrările cu nume lung din directoare sunt adăugate pe măsură ce se creează fișiere/directoare cu nume lung.

Intrările cu nume lung sunt ca fișierele ascunse sau ca și cele ascunse sau sistem, pentru sistem *down-level*. Acestea sunt suficiente pentru a nu cauza probleme utilizatorului obișnuit. Utilizatorul poate muta fișierele folosind formatul 8.3 pe alt disc și invers, fără efecte secundare.

Este interesant ce se întâmplă pe disc cu un sistem de fișiere *FAT down-level* în momentul în care este schimbat directorul curent. Acest lucru poate afecta intrările cu nume lung, deoarece sistemele *down-level* ignoră aceste intrări și nu se asigură dacă sunt asociate corect numelor scurte.

Un sistem *down-level* vede aceste intrări numai atunci când caută eticheta de volum a discului. Pe un sistem *down-level*, eticheta de volum poate fi returnată incorect în cazul în care aceasta nu apare înaintea tuturor intrărilor cu nume lung în directorul rădăcină. Acest lucru se datorează faptului că aceste intrări au atributul de volum setat.

Dacă se încearcă ștergerea etichetei de volum, s-ar putea ca o intrare cu nume lung să fie ștearsă. Numele lung nu va mai fi valid deoarece unele dintre intrările sale sunt marcate ca fiind șterse.

Dacă un fișier este redenumit pe un sistem *down-level*, atunci numai numele scurt va fi redenumit. În acest caz numele lung nu va fi afectat, dar datorită faptului că prin redenumire se schimbă suma de control, intrările cu nume lung vor deveni invalide. Această redenumire nu trebuie să intre în conflict cu nici un nume lung. Altfel, un sistem *down-level* poate crea un nume scurt într-un fișier care este identic cu un nume al altui fișier, atunci când nu se ține seama de tipul literelor. Dacă fișierul este șters, atunci numele lung al acestuia, este un simplu orfan. Dacă sunt create câteva fișiere noi, numele lung poate fi incorect asociat cu noul fișier. Ca și în cazul redenumirii, suma de control ajută la prevenirea acestor asocieri.

Alte informații cu privire la directoarele FAT

Intrările în director cu nume lung, sunt identice pe toate tipurile de *FAT*. Pentru volumele *FAT32*, *driver*-ul sistemului de fișiere nu trebuie să permită crearea unui lanț de *cluster*-e mai mare de `0x100000000` bytes, și ultimul byte al lanțului care are această mărime nu poate fi alocat fișierului. Din acest motiv, nici un fișier nu poate avea mărimea mai mare decât `0xFFFFFFFF` bytes. Aceasta este limita fundamentală pe toate tipurile de *FAT*.

De asemenea, *driver*-ul unui sistem de fișiere *FAT* nu trebuie să permită unui director (un fișier care este de fapt un *container* pentru alte fișiere) să fie mai mare de `65536 * 32 = 2.097.152` bytes.

Această limită nu se aplică numărului de fișiere din director, ci reprezintă mărimea directorului și nu are nimic în comun cu conținutul directorului. Există două motive întemeiate pentru această limită:

1. Deoarece directoarele nu sunt sortate sau indexate, nu este recomandabilă crearea de directoare uriașe (care conțin foarte multe fișiere). În cazul directoarelor mari, operațiile cum ar fi cea de adăugare (care verifică toate intrările din director, pentru a vedea dacă intrarea ce urmează a fi creată mai există sau nu) sunt foarte ineficiente.
2. Sunt multe *driver*-e pentru sistemele de fișiere *FAT* și utilitare de disc, inclusiv cele de la *Microsoft*, care nu sunt capabile să citească atât de multe intrări în director, deoarece numărul de intrări din director este stocat folosind o variabilă reprezentată pe 16 biți.

Bibliografie

• www.microsoft.com

Ciprian Ileană și Claudiu Soroiu sunt studenți în anul I la Universitatea Babeș-Bolyai din Cluj. Pot fi contactați la adresele ciprian_il@yahoo.com, *respectiv* csoroiu@yahoo.com.