Sisteme de FIȘIERE

Ciprian Ileană, Claudiu Soroiu

În cadrul acestui articol vom prezenta sistemul de fișiere folosit de sistemele de operare produse de Microsoft. Vom începe cu sistemul FAT 12 și vom continua cu FAT 16 și FAT 32.

Introducere

Sistemul de fișiere ***FAT*** (***F**ile **A**llocation **T**able*) a apărut la sfârșitul anilor '70, fiind sistemul de fișiere suportat de sistemul de operare *MS-DOS*.

Inițial, a fost dezvoltat ca un simplu sistem de fișiere pentru *floppy disk*-uri cu dimensiune mai mică de 500 KB. Cu timpul a fost îmbunătățit, pentru a suporta medii mai mari de date. În prezent sunt trei tipuri de *FAT*: *FAT12*, *FAT16* și *FAT32*. Diferența de bază între aceste trei tipuri de *FAT* este aceea (de unde și numele) că intrările fișierelor din *FAT* au mărimi diferite ca număr de biți. Într-o intrare de *FAT12* sunt folosiți 12 biți, pentru *FAT16* sunt folosiți 16 biți iar pentru *FAT32* se folosesc 32 de biți.

Privire de ansamblu

La origine, toate sistemele de fișiere *FAT* au fost dezvoltate pentru PC-uri cu arhitecturi compatibile *IBM*. În consecință, toate structurile de date sunt "*little endian*" (datele sunt scrise în format ***LSB** - **L**ess **S**ignificant **B**yte First*). De exemplu, o intrare de *FAT32* este stocată pe disc ca o serie de patru bytes (1 byte = 8 biți), primul fiind byte[0] iar ultimul byte[3].

În cele ce urmează se observă numerotarea celor 32 de biți începând cu bitul 0 (0 este cel mai nesemnificativ bit - dreapta jos).

```
31 30 29 28 27 26 25 24 - byte-ul 3
23 22 21 20 19 18 17 16 - byte-ul 2
15 14 13 12 11 10 09 08 - byte-ul 1
07 06 05 04 03 02 01 00 - byte-ul 0
```

Este important dacă un anumit sistem este unul "*big endian*" (cum sunt sistemele livrate de *Motorola*), deoarece în timpul transferurilor de date intervine transformarea acestora din ***MSB*** (***M**ost **S**ignificant **B**yte First*) în ***LSB*** și invers.

Un sistem de fișiere *FAT* este compus din patru zone de bază, care sunt aranjate în ordinea care urmează:

- zona rezervată
- zona *FAT* (tabelele de alocare a fișierelor)
- zona directorului rădăcină (*ROOT* - a fost eliminată începând cu *FAT32*)
- zona de date a fișierelor și directoarelor (*FDDR*).

Zona Rezervată

Prima structură de date importantă pe un volum *FAT* este numită ***BPB*** (***BI**OS **P**arameter **B**lock*) și este localizată în primul sector al volumului (partiției), în zona rezervată. Acest sector este numit uneori "*boot sector*" (sector de *boot*) sau "*sectorul 0*", dar lucrul cel mai important este că acesta este primul sector al volumului (partiției).

Acesta este un prim element care cauzează uneori confuzie. În *MS-DOS* versiunea `1.x` nu exista *BPB* în sectorul de *boot*. În această primă versiune a sistemului de fișiere *FAT*, existau doar două formate diferite: unul pentru *floppy disk*-uri cu o singură față și unul pentru *floppy disk*-uri cu două fețe (de 360K - 5.25"). Determinarea tipului era realizată pe baza primului byte din *FAT* (primii 8 biți ai lui FAT[0]).

Acest mod de determinare a formatului a fost eliminat începând cu versiunea `2.x` prin adăugarea unui *BPB* în sectorul de *boot*. Astfel, vechiul stil de determinare a fost complet eliminat. În consecință toate volumele (partițiile) *FAT* trebuie să aibă un *BPB* în sectorul de boot.

Acest fapt duce la o nouă (posibilă) confuzie, relativ la determinarea tipului de *FAT*.

În continuare vom prezenta această structură de date. *BPB*-ul aflat în sectorul de boot definit în *MS-DOS* `2.x` permitea doar dimensiuni de cel mult 65535 sectoare (mai puțin de 32 MB în sectoare de câte 512 bytes). Această limitare se datora faptului că numărul total de sectoare era un câmp reprezentat pe 16 biți.

Începând cu *MS-DOS* `3.x` s-a modificat *BPB*-ul astfel încât să includă un câmp reprezentat pe 32 de biți, pentru a reține numătul total de sectoare.

Următoarea schimbare a *BPB*-ului a apărut la sistemul de operare *Windows '95 OSR2*, când a fost introdus *FAT32*. *FAT16* era limitat de mărimea maximă a *FAT*-ului și de mărimea maximă a unui *cluster* valid la nu mai mult de 2 GB dacă discul hard conținea sectoare de 512 bytes. *FAT32* elimină această limitare și astfel volumul (partiția) poate ocupa mai mult de 2 GB.

Primele câmpuri din *BPB*-ul *FAT32* (până la `BPB_TotSec32` inclusiv) sunt identice cu cele din *BPB*-ul *FAT12/FAT16*. Ele încep să difere de la offset-ul 36, în funcție de tipul de *FAT* folosit (vezi determinarea tipului de FAT).

Important este că *BPB*-ul aflat în sectorul de *boot* al unui *FAT* trebuie să conțină toate câmpurile noi ce apar față de *BPB*-ul *FAT12/FAT16* sau *FAT32*. Acest lucru asigură compatibilitatea unui program (*driver*) care trebuiie să înțeleagă și să suporte volumele corect, deoarece conține întotdeauna toate câmpurile definite.

În continuare vom prezenta sectorul de *boot*. Toate câmpurile al căror nume începe cu `BPB_` aparțin *BPB*-ului. Câmpurile al căror nume încep cu `BS_` fac parte din sectorul de *boot* și nu din `BPB`.

Primul câmp este `BS_jmpBoot`, apare la *offset*-ul 0 și are dimensiunea de trei bytes. El conține o instrucțiune de salt la codul de *boot*. Acest câmp poate avea două forme:

```
JmpBoot[0]=0x0EB,
jmpBoot[1]=0x??,
jmpBoot[2]=0x90
```

și:

```
JmpBoot[0]=0x0E9,
jmpBoot[1]=0x??,
jmpBoot[2]=0x??
```

Prin `0x??` se indică faptul că byte-ul respectiv poate conține orice valoare care poate fi reprezentată folosind opt biți.

Al doilea câmp este `BS_OEMName`, apare la *offset*-ul 3 și are dimensiunea de opt bytes. Acest câmp conține șirul `MSWIN4.1`. Sunt multe interpretări greșite ale acestui câmp; sistemele de operare de la *Microsoft* nu acordă atenție acestui câmp, în timp ce unele *driver*-e de FAT îl iau în considerare. Acesta este motivul pentru care șirul de caractere indicat este `MSWIN4.1`.

Al treilea câmp este `BPB_BytsPerSec`, apare la *offset*-ul 11 și are dimensiunea de doi bytes. Este folosit pentru a indica numărul de bytes care fac parte dintr-un sector. Valorile admise sunt 512, 1024, 2048 și 4096.

Al patrulea câmp este `BPB_SecPerClus`, apare la *offset*-ul 13 și are dimensiunea de un byte. Valoarea sa indică numărul de sectoare dintr-o unitate de alocare (*cluster*). Valoarea sa trebuie să fie o putere a lui 2, valorile permise fiind 1, 2, 4, 8, 16, 32, 64 și 128. Trebuie amintit faptul că că un *cluster* are dimensiunea (în bytes) dată de produsul `BPB_BytsPerSec*BPB_SecPerClus`. Cea mai mare valoare permisă pentru mărimea unui *cluster* de 64 KB.

Al cincilea câmp este `BPB_RsvdSecCnt`, apare la *offset*-ul 14 și are dimensiunea de doi bytes. Valoarea sa indică numărul de sectoare din zona rezervată a volumului (partiției). Acest câmp nu trebuie să fie 0. Pentru *FAT12* și *FAT16* această valoare nu trebuie să fie diferită de 1, iar pentru *FAT32*, ea este de obicei 32.

Al șaselea câmp este `BPB_NumFATs`, apare la *offset*-ul 16 și are dimensiunea de un byte. El indică numărul de structuri *FAT* prezente pe volumul curent. Acest câmp trebuie să conțină întotdeauna valoarea 2 pentru orice volum de tip *FAT*. Cu toate acestea, orice valoare mai mare sau egală cu 1 este permisă. Foarte puține programe și sisteme de operare nu funcționează corect dacă sunt prezente structuri de *FAT* în număr diferit de 2.

Al șaptelea câmp este `BPB_RootEntCnt`, apare la *offset*-ul 17 și are dimensiunea de doi bytes. Pentru *FAT12* și *FAT16*, acest câmp conține numărul maxim de intrări permise în directorul rădăcină. Pentru *FAT32*, acest câmp trebuie să aibă valoarea 0. În cazul volumelor *FAT12* și *FAT16* acest câmp trebuie să aibă o valoare care, înmulțită cu 32, este un multiplu al valorii câmpului `BPB_BytsPerSec`. Pentru compatibilitate maximă, volumele *FAT16* ar trebui să folosească valoarea 512.

Al optulea câmp este `BPB_TotSec16`, apare la *offset*-ul 19 și are dimensiunea de doi bytes. Acest câmp conține numărul total de pe toate cele patru regiuni ale volumului. Dacă valoarea sa este 0, atunci valoarea câmpului `BPB_TotSec32` nu poate fi tot 0. Dacă valoarea acestui al doilea câmp este 0, atunci înseamnă că numărul total al sectoarelor este mai mic decât 65536, deci poate fi reprezentat folosind câmpul `BPB_TotSec16` care conține doar 16 biți.

Al nouălea câmp este `BPB_Media`, apare la *offset*-ul 21 și are dimensiunea de un byte. Pentru discurile fixe valoarea standard este `0xF8`. Pentru discurile mobile este folosită adesea valoarea `0xF0`. Valorile admise pentru acest câmp sunt `0xF0`, `0xF8`, `0xF9`, `0xFA`, `0xFB`, `0xFC`, `0xFD`, `0xFE` și `0xFF`. Singurul lucru important este acela că această valoare trebuie să apară în byte-ul cel mai nesemnificativ al primei intrări în *FAT*, `FAT[0]`.

Al zecelea câmp este `BPB_FATSz16`, apare la *offset*-ul 22 și are dimensiunea de doi bytes. Acest câmp reprezintă numărul de sectoare ocupate de un *FAT* pentru volumele *FAT12/FAT16*. Pe volumele *FAT32* acest câmp trebuie să conțină valoarea 0; câmpul `BPB_FATSz32` va conține mărimea în sectoare a unui *FAT*.

Al unsprezecelea câmp este `BPB_SecPerTrk`, apare la *offset*-ul 24 și are dimensiunea de doi bytes. Acest câmp conține numărul de sectoare pe pistă și este folosit de întreruperea `0x13`.

Al doisprezecelea câmp este `BPB_SecPerTrk`, apare la *offset*-ul 26 și are dimensiunea de doi bytes. Întreruperea `0x13` folosește acest câmp pentru a determina numărul de capete. Pentru o dischetă de 1.44 MB, valoarea sa este 2.

Al treisprezecelea câmp este `BPB_SecPerTrk`, apare la *offset*-ul 28 și are dimensiunea de patru bytes. Acest câmp conține numărul de sectoare ascunse care preced

partiția care conține acest volum *FAT*. Valoarea sa trebuie să fie 0 pentru medii care nu sunt partiționate.

Al paisprezecelea câmp este `BPB_TotSec32`, apare la *offset*-ul 32 și are dimensiunea de patru bytes. Acesta conține numărul de sectoare ale tuturor celor patru zone ale volumului. Valoarea sa poate fi 0; în acest caz câmpul `BPB_TotSec16` trebuie să aibă valoare diferită de 0. Pentru volumele *FAT32* valoarea acestui câmp este diferită de 0. Pentru volumele *FAT12/FAT16* acest câmp conține numărul de sectoare în cazul în care numărul total de sectoare depășește valoarea 65535; în acest caz valoarea câmpului `BPB_TotSec16` este 0.

Din acest punct (*offset*-ul 36) structura *BPB/boot* diferă pentru *FAT12/FAT16* și *FAT32*. Vom prezenta acum structura pentru volumele *FAT12* și *FAT16* începând cu *offset*-ul 36, al sectorului de *boot*.

Câmpul `BS_DrvNum` apare la *offset*-ul 36 și are dimensiunea de un byte. Întreruperea `0x13` folosește acest câmp pentru a determina o valoare caracteristică *drive*-ului. Pentru discuri hard valoarea este cuprinsă între `0x80` și `0x83` iar pentru dischete între `0x00` și `0x01`.

Câmpul `BS_Reserved1` apare la *offset*-ul 37 și are dimensiunea de un byte. Acest câmp este rezervat (folosit la formatare de *Windows NT*) și trebuie să aibă valoarea 0.

Câmpul `BS_BootSig` apare la *offset*-ul 38 și are dimensiunea de un byte. Valoarea sa este constanta `0x29` și indică prezența următoarelor trei câmpuri.

Câmpul `BS_VolID` apare la *offset*-ul 39 și are dimensiunea de patru bytes. Valoarea acestui câmp reprezintă numărul serial (*serial number*) al volumului. Este folosit, în general, pentru medii care nu sunt fixe, cum ar fi dischetele.

Câmpul `BS_VolLab` apare la offset-ul 43 și are dimensiunea de 11 bytes. El este folosit pentru a păstra eticheta de volum.

Câmpul `BS_FilSysType` apare la offset-ul 54 și are dimensiunea de opt bytes. Acest câmp conține unul din șirurile de caractere: `"FAT12   "`, `"FAT16   "` sau `"FAT     "` (fără ghilimele). Mulți cred că acest șir de caractere are legatură cu tipul de *FAT* folosit pe acest volum; acest lucru nu este însă adevărat.

Vom prezenta acum structura sistemului *FAT32* începând cu *offset*-ul 36.

Câmpul `BPB_FATSz32` apare la *offset*-ul 36 și are dimensiunea de patru bytes. Acest câmp este definit numai pentru *FAT32* și nu există pe volume *FAT12* și *FAT16*. El conține numărul de sectoare ocupate de un singur *FAT*. Valoarea câmpului `BPB_FATSz16` trebuie neapărat să fie 0.

Câmpul `BPB_ExtFlags` apare la *offset*-ul 40 și are dimensiunea de doi bytes. Acest câmp este și el definit numai pe volumele *FAT32*. Biții 0-3 indică numărul de structuri *FAT* active. Acest număr este valid dacă *mirroring*-ul este dezactivat. Biții 4-6 sunt rezervați. Dacă bitul are valoarea 0, înseamnă că *FAT*-ul este oglindit în toate FAT-urile definite. Biții 8-15 sunt rezervați.

Câmpul `BPB_FSVer` apare la *offset*-ul 42 și are dimensiunea de doi bytes. Acest câmp este, de asemenea, definit numai pe volumele *FAT32.* Byte-ul cel mai semnificativ reprezintă numărul major al reviziei și byte-ul cel mai puțin semnificativ reprezintă numărul minor al reviziei. Acest câmp reprezintă versiunea volumului *FAT32*. Versiunea curentă este 0:0. Dacă această valoare este diferită de 0, versiunile de *Windows* precedente nu vor monta acest volum.

Câmpul `BPB_RootClus` apare la *offset*-ul 44 și are dimensiunea de patru bytes. Acest câmp, definit numai pentru *FAT32*, reprezintă numărul *cluster*-ului la care începe directorul rădăcină.

Câmpul `BPB_FSInfo` apare la *offset*-ul 48 și are dimensiunea de doi bytes. Este și el definit numai pentru *FAT32* și reprezintă numărul sectorului unde se află structura *FSINFO*, în zona rezervată. De obicei acest câmp are valoarea 1.

Câmpul `BPB_BkBootSec` apare la *offset*-ul 50 și are dimensiunea de doi bytes. De asemenea, acest câmp este definit numai pe volumele *FAT32*; dacă acest câmp are valoarea diferită de 0, atunci reprezintă numărul sectorului din zona rezervată unde se află o copie a sectorului de *boot*. De obicei această valoare este 6; nu se recomandă folosirea unei alte valori.

Câmpul `BPB_Reserved` apare la *offset*-ul 52 și are dimensiunea de 12 bytes. Acesta este definit numai pe volumele *FAT32* și este rezervat pentru expansiunea viitoare a acestei structuri *FAT32*.

Câmpul `BS_DrvNum` apare la *offset*-ul 64 și are dimensiunea de un byte. Acest câmp are aceeași semnificație ca și pentru *FAT12* și *FAT16*. Singura diferență este faptul că pentru volumele FAT32 acest câmp apare la un alt *offset*.

Câmpul `BS_Reserved1` apare la *offset*-ul 65 și are dimensiunea de un byte. Acest câmp are și el aceeași semnificație ca și pentru *FAT12* și *FAT16*, singura diferență fiind, și în acest caz, faptul că pentru volumele *FAT32* apare la un alt *offset*.

Câmpul `BS_BootSig` apare la *offset*-ul 66 și are dimensiunea de un byte. Acest câmp are, de asemenea, aceeași semnificație ca și pentru *FAT12* și *FAT16*, dar apare la un alt *offset*.

Câmpul `BS_VolID` apare la *offset*-ul 67 și are dimensiunea de patru bytes. Acesta are aceeași semnificație ca și pentru *FAT12* și *FAT16*, singura diferență fiind *offset*-ul la care apare.

Câmpul `BS_VolID` apare la *offset*-ul 71 și are dimensiunea de 11 bytes. Din nou, semnificația este identică celei de la sistemele *FAT12* și *FAT16*, singura diferență fiind, și de această dată, *offset*-ul la care apare.

Câmpul `BS_FilSysType` apare la *offset*-ul 82 și are dimensiunea de opt bytes. Pentru sistemele *FAT32* valoarea sa este `"FAT32   "` (fără ghilimele), dar, la fel ca și în cazul sistemelor *FAT12* și *FAT16*, această valoare nu are nici o relevanță.

Mai trebuie adăugat un lucru în ceea ce privește sectorul 0 al unui volum *FAT*. Dacă am considera conținutul

sectorului ca un tablou de bytes (începând cu indicele 0), ar trebui ca `Sector[510]` să aibă valoarea `0x55` și `Sector[511]` să aibă valoarea cu `0xAA`.

Multe documentații referitoare la FAT afirmă, greșind bineînțeles, că semnătura `0xAA55` ocupă "*ultimii doi bytes ai sectorului de boot*". Această afirmație este corectă numai dacă valoarea câmpului `BPB_BytsPerSec` este 512. Dacă această valoare este mai mare decât 512, *offset*-ul acestei semnături nu se schimbă (cu toate că nu este greșit să se pună această semnătură și la sfârșitul sectorului).

Să presupunem că avem o partiție cu mărimea în sectoare `DskSz`. Este important să se verifice valoarea câmpului `BPB_TotSec16/32` (unul dintre cele două câmpuri este diferit de zero, cel diferit de zero se verifică); această valoare nu poate fi mai mare decât `DskSz`. Valoarea acestor câmpuri trebuie să fie mai mică sau egală cu `DskSz`, deoarece nu este nimic în neregulă cu volumul *FAT*.

De fapt, nu este nimic neobișnuit să avem `BPB_TotSec16/32` cu valoarea mai mică decât `DskSz`. Este în regulă dacă `BPB_TotSec16/32` este considerabil mai mic decât `DskSz`.

Aceasta înseamnă că se irosește spațiu pe disc și nu faptul că volumul *FAT* este alterat în vreun fel. Totuși, dacă `BPB_TotSec16/32` este mai mare decât `DskSz`, volumul este serios distrus sau malformat deoarece el se întinde peste limita discului sau partiția se suprapune peste datele care urmează pe disc, după sfârșitul ei. Dacă un volum pentru care `BPB_TotSec16/32` este prea mare pentru discul pe care apare este considet valid, pot apărea pierderi catastrofale de date.

Zona FAT

Următoarea structură importantă este *FAT*-ul însuși. Această structură de date definește o listă simplu înlănțuită a *cluster*-elor unui fișier. Trebuie remarcat faptul că, în acest punct, directorul din *FAT* sau un *container* de fișier nu sunt altceva decât simple fișiere. Singurul lucru care face diferența dintre un director și un fișier este acela că directorul are un atribut special care indică faptul că "*fișierul*" este director. Următorul lucru important despre directoare este că "*fișierul*" conține o serie de intrări în *FAT* și ocupă un spațiu de 32 de bytes. În toate celelalte privințe, directorul este similar unui fișier.

FAT-ul unui anumit volum este o hartă numerotată a *cluster*-elor, unde primul *cluster* de date este *cluster*-ul 2.

Primul sector al *cluster*-ului 2 (regiunea de date a discului) este stocat folosind câmpurile *BPB* pentru un volum. Pentru început trebuie determinată mărimea directorului rădăcină în număr de sectoare :

```
RootDirSectors=((BPB_RootEntCnt*32)+
 (BPB_BytsPerSec-1))/BPB_BytsPerSec;
```

Pe un volum *FAT32* câmpul `BPB_RootEntCnt` are întotdeauna valoarea `0`. Numărul `32` din formula precedentă reprezintă mărimea în bztes a unei intrări din *FAT* corespunzătoare unui director. A se observa, de asemenea, că rezultatul acestui calcul se rotunjește prin adaos.

Începutul zonei de date, primul sector al *cluster*-ului 2, se calculează în felul următor:

```
if(BPB_FATSz16!=0)
  FATSz=BPB_FATSz16;
else
  FATSz=BPB_FATSz32;
FirstDataSector=BPB_ResvdSecCnt+
  (BPB_NumFATs*FATSz)+RootDirSectors;
```

De observat că numărul acestui sector este relativ la primul sector al volumului care conține *BPB*-ul (sectorul care conține *BPB*-ul este sectorul cu numărul 0). Acesta nu trebuie să fie situat relativ la sectorul considerat ca fiind începutul partiției, deoarece, după partiționare, sectorul 0 al partiției nu este neapărat și sectorul 0 al *drive*-ului.

Fiind dat numărul *N* al unui *cluster* de date valid, numărul primului sector al *cluster*-ului (relativ la sectorul 0 al volumului *FAT*) se calculează în felul următor.

```
FirstSectorofCluster=((N-2)*
  BPB_SecPerClus)+FirstDataSector;
```

Deoarece `BPB_SecPerClus` este restricționat la puteri ale lui 2 (1, 2, 4, 8, 16, 32, ...), acest lucru înseamnă că înmulțirile și împărțirile cu `BPB_SecPerClus` pot fi realizate cu ajutorul deplasărilor pe biți care sunt mult mai rapide comparativ cu instrucțiunile `MUL` și `DIV`. Procesoarele cu arhitectură *Intel x86* execută cu aceeași viteză o instrucțiune `MUL` sau `DIV` cu o putere a lui 2 ca și o deplasare pe biți cu puterea lui 2.

Determinarea tipului de FAT

Apar foarte mari confuzii asupra modului exact de determinare a tipului de *FAT* care duc la erori de genul "*depășire cu* 1", "*depășire cu* 2", "*depășire cu* 10" sau "*depășire masivă*".

De fapt, modul de determinare este unul foarte simplu. Tipul de *FAT* (*FAT12*, *FAT16* sau *FAT32*) este determinat folosind doar numărul de *cluster*-e ale partiției și nimic altceva.

În continuare este recomandată citirea cu atenție a tuturor termenilor deoarece au o foarte mare importanță. De exemplu, termenul "*numărul de cluster-e*" este diferit de termenul "*numărul maxim de cluster-e valide*" deoarece primul *cluster* de date este 2, nu 0 sau 1.

Pentru început să vedem cum este determinată valoarea "*numărului de cluster-e*". Această valoare se determină cu ajutorul câmpurilor din *BPB*-ul partiției.

Pentru început se determină numărul de sectoare ocupate de directorul rădăcină, după cum s-a arătat anterior:

```
RootDirSectors=((BPB_RootEntCnt*32)+
  (BPB_BytsPerSec-1))/BPB_BytsPerSec;
```

Trebuie remarcat faptul că pe un volum *FAT32* câmpul `BPB_RootEntCnt` are întotdeauna valoarea `0`.

În continuare vom determina numărul de sectoare din regiunea de date a partiției:

```
if (BPB_FATSz16!=0)
  FATSz=BPB_FATSz16;
else
  FATSz=BPB_FATSz32;
if (BPB_TotSec16!=0)
  TotSec=BPB_TotSec16;
else
  TotSec=BPB_TotSec32;
DataSec=TotSec-(BPB_ResvdSecCnt+
  BPB_NumFATs*FATSz+RootDirSectors);
```

Acum vom determina numărul de *cluster*-e:

```
CountofClusters=DataSec/
  BPB_SecPerClus;
```

A se observa că acest calcul se rotunjește prin lipsă.

În acest moment putem determina tipul de *FAT*. Operațiile trebuie efectuate cu atenție deoarece este foarte posibil să apară o eroare de tipul "*depășire cu* 1".

În exemplul ce urmează vom descrie modul de determinare al tipului de *FAT*. Acolo unde se întâlnește semnul `"<"` nu înseamnă și `"<="`. Toate numerele de mai jos sunt corecte: primul număr (pentru *FAT12*) este 4085; al doilea număr (pentru *FAT16*) este 65525.

```
if (CountofClusters<4085){
  /* Volum FAT12 */
}
else
  if (CountofClusters<65525){
    /* Volum FAT16 */
  }
  else{
    /* Volum FAT32 */
  }
```

Aceasta este singura metodă de determinare a tipului de *FAT*. Nu există volume *FAT12* cu mai mult de 4084 *cluster*-e, nici volume (partiții) *FAT16* cu mai puțin de 4085 *cluster*-e sau mai mult de 65524 *cluster*-e precum nici volume (partiții) de *FAT32* cu mai puțin de 65525 *cluster*-e. Dacă veți încerca să realizați o partiție *FAT* care încalcă aceste reguli, sistemele de operare *Microsoft* nu vor funcționa corect deoarece vor considera că este vorba de un tip diferit de *FAT* față de cel considerat de dumneavoastră.

Se observă că valoarea `CountofClusters` este exact cea calculată mai sus. Cel mai mare număr al unui *cluster* de date valid pentru un volum este `CountOfClusters+1` și "*numărul total de cluster-e inclusiv cele două cluster-e rezervate*" este `CountofClusters+2`.

Mai avem un singur calcul important legat de *FAT*. Fiind dat numărul *N* al unui *cluster* valid, se va calcula unde este localizată în *FAT* intrarea pentru acest *cluster*. Acest lucru se calculează foarte ușor pentru *FAT16* și *FAT32*, calculul complicându-se pentru *FAT12*:

```
if (BPB_FATSz16!=0)
  FATSz=BPB_FATSz16;
else FATSz=BPB_FATSz32;
if (FATType==FAT16)
  FATOffset=N*2;
else
  if (FATType==FAT32)
    FATOffset=N*4;
ThisFATSecNum=BPB_ResvdSecCnt+
  (FATOffset/BPB_BytsPerSec);
ThisFATEntOffset=FATOffset%
  BPB_BytsPerSec;
```

În `ThisFATSecNum` este reținut numărul sectorului din primul *FAT* care conține intrarea pentru al *N*-lea *cluster* la *offset*-ul `ThisFATEntOffset`. Dacă se dorește calcularea numărului sectorului din al doilea *FAT*, se va aduna `FATSz` la `ThisFATSecNum`, iar pentru al treilea *FAT* se va aduna `2*FATSz` ș.a.m.d.

În acest moment se poate citi sectorul `ThisFATSecNum` (numărul acestui sector este relativ la sectorul `0` al volumului *FAT*). Considerăm că acest sector se citește folosind un tablou de bytes numit `SecBuff` și că `WORD` reprezintă un tip de date fără semn pe 16 biți, iar `DWORD` este un tip de date fără semn pe 32 de biți.

```
if (FATType==FAT16)
  FAT16ClusEntryVal=*((WORD*)
    &SecBuff[ThisFATEntOffset]);
else
  FAT32ClusEntryVal=(*((DWORD*)
    &SecBuff[ThisFATEntOffset]))
    &0x0FFFFFFF;
```

Modificarea conținutului unui *cluster* în *FAT16/32* se face în felul următor:

```
if (FATType==FAT16)
  *((WORD*)&SecBuff[ThisFATEntOffset])
    =FAT16ClusEntryVal;
else{
  FAT32ClusEntryVal=FAT32ClusEntryVal
    &0x0FFFFFFF;
  *((DWORD*)&SecBuff[ThisFATEntOffset])
  =(*((DWORD*)&SecBuff[ThisFATEntOffset]))
  &0xF0000000;
  *((DWORD*)&SecBuff[ThisFATEntOffset])
  =(*((DWORD*)&SecBuff[ThisFATEntOffset]))
  |FAT32ClusEntryVal;
}
```

Vom descrie modul de funcționare a codului de mai sus pentru *FAT32*. O intrare *FAT* pentru *FAT32* ocupă, de fapt, 28 de biți. Cei mai semnificativi patru biți ai unei intrări în *FAT* sunt rezervați. Cei patru biți trebuie modificați când se formatează volumul *FAT32* și atunci toate intrările în *FAT32* au cei mai semnificativi patru biți setați pe 0.

Aici trebuie dată o explicație suplimentară, pentru a evita apariția confuziilor. Pentru început trebuie reținut faptul că intrările pe 32 de biți în *FAT* sunt, de fapt, intrări pe 28 de biți. De exemplu, toate cele trei intrări pe 32 de biți `0x10000000`, `0xF0000000` și `0x00000000`, indică faptul că acel *cluster* este liber, deoarece se ignoră cei mai semnificativi patru biți când se citește valoarea intrării *cluster*-ului.

Dacă valoarea pe 32 de biți a valorii *cluster*-ului liber este `0x30000000` și doriți să îl marcați ca fiind un *cluster* "*bad*" atunci se va scrie valoarea `0x0FFFFFF7`. Intrarea pe 32 de biți va conține valoarea `0x3FFFFFF7`, deoarece trebuie să păstrați cei mai semnificativi patru biți, în momentul în care se scrie intrarea `0x0FFFFFF7` care reprezintă marcajul pentru un *cluster* "*bad*".

Deoarece `BPB_BytsPerSec` este întotdeauna divizibil cu 4, nu trebuie să ne îngrijoreze faptul că o intrare *FAT16/32* ar putea depăși un sector, deoarece acest lucru nu se poate întâmpla.

Codul sursă pentru *FAT12*, este mai complicat pentru că se folosesc 1,5 bytes (12 biți) pentru o intrare în *FAT*.

```
if (FATType==FAT12)
  FATOffset=N+(N/2);
ThisFATSecNum=BPB_ResvdSecCnt+
  (FATOffset/BPB_BytsPerSec);
ThisFATEntOffset=REM(FATOffset/
  BPB_BytsPerSec);
```

Acum trebuie verificat cazul în care se produce depășirea unui sector:

```
if (ThisFATEntOffset==(BPB_BytsPerSec
                                  -1)){
// depasire de sector
}
```

În acest moment se accesează intrarea în *FAT* ca `WORD`, la fel ca pentru *FAT16*, dar dacă numărul *cluster*-ului este par atunci avem nevoie numai de cei mai puțin semnificativi 12 biți ai intrării de 16 biți pe care o prelucrăm; în cazul în care numărul *cluster*-ului este impar, atunci avem nevoie de cei mai semnificativi 12 biți din cei 16 cu care lucrăm.

```
FAT12ClusEntryVal=*((WORD*)
  &SecBuff[ThisFATEntOffset]);
if (N&0x0001)
  FAT12ClusEntryVal=
    FAT12ClusEntryVal>>4;
else
  FAT12ClusEntryVal=
    FAT12ClusEntryVal&0x0FFF;
```

Pentru a seta conținutul aceluiași *cluster* se folosește următorul cod:

```
if (N&0x0001){
  FAT12ClusEntryVal=
    FAT12ClusEntryVal<<4;
  *((WORD*)&SecBuff[ThisFATEntOffset])
    =(*((WORD*)&SecBuff[
    ThisFATEntOffset]))&0x000F;
}
else{
  FAT12ClusEntryVal
    =FAT12ClusEntryVal&0x0FFF;
  *((WORD*)&SecBuff[ThisFATEntOffset])
    =(*((WORD*)&SecBuff[
    ThisFATEntOffset]))&0xF000;
}
*((WORD*)&SecBuff[ThisFATEntOffset])
  =(*((WORD*)&SecBuff[
  ThisFATEntOffset]))
  |FAT12ClusEntryVal;
```

Operatorul `">>"` realizează deplasarea la dreapta pe biți și cei mai semnificativi patru biți primesc valoarea 0; operatorul `"<<"` realizează deplasarea la stânga pe biți și cei mai puțin semnificativi patru biți primesc valoarea 0.

Modul în care datele unui fișier sunt asociate cu fișierul este realizat astfel: în intrarea în director a fișierului/directorului este stocat numărul primului *cluster* de la care începe fișierul; primul *cluster* al fișierului este asociat cu datele conținute de primul *cluster* din fișier și localizarea acestor date pe partiție (volum) se determină folosind numărul *cluster*-ului.

Un fișier de dimensiune 0 (fișier care nu are alocate în nici un fel date) are numărul primului *cluster* 0, plasat în propria intrare din director. Locația acestui *cluster* în *FAT* conține fie marcajul de sfârșit de fișier (***EOC*** - ***End Of Clusterchain***), fie numărul următorului *cluster* din fișier. Valoarea *EOC* depinde de tipul *FAT*-ului (să presupunem că `FATContent` este conținutul intrării *cluster*-ului în *FAT* care va fi verificată dacă este sau nu *EOC*):

```
IsEOF=FALSE;
if (FATType==FAT12){
  if (FATContent>=0x0FF8)
    IsEOF=TRUE;
}
else
  if (FATType==FAT16){
    if (FATContent>=0xFFF8)
      IsEOF=TRUE;
  }
```

```
else
  if (FATType==FAT32){
    if (FATContent>=0x0FFFFFF8)
      IsEOF=TRUE;
  }
```

Numărul *cluster*-ului a cărui intrare în *FAT* conține *EOC* este alocat fișierului și este de asemenea ultimul *cluster* alocat fișierului.

Sistemele de operare *Microsoft* folosesc *driver*-e care utilizează pentru *EOC* valorile `0x0FFF` pentru *FAT12*, `0xFFFF` pentru *FAT16* și `0x0FFFFFFF` pentru *FAT32* când setează conținutul unui *cluster* din *FAT* cu valoarea *EOC*.

Există mai multe utilitare de disc pentru sistemele de operare *Microsoft* care setează sfârșitul de fișier cu altă valoare (între `0x0FF8` și `0x0FFF` pentru *FAT12*; `0xFFF8` și `0xFFFF` pentru *FAT16*; `0x0FFFFFF8` și `0x0FFFFFFF` pentru *FAT32*).

Mai există un marcaj special folosit pentru "*Bad Clusters*" (*cluster*-e avariate). Orice *cluster* care conține valoarea acestui marcaj nu va fi folosit pe post de *cluster* liber deoarece poate conduce la erori fatale ale discului. Acest marcaj are valorile: `0x0FF7` pentru *FAT12*, `0xFFF7` pentru *FAT16* și `0x0FFFFFF7` pentru *FAT32*.

Un alt lucru foarte important este că *bad*-urile sunt *cluster*-e "*pierdute*" - *cluster*-e care par să fie alocate deoarece nu conțin valoarea 0, dar care nu aparțin lanțului de alocare al nici unui fișier.

Observați că nu este posibil ca un *cluster* al cărui marcaj este "*bad*" să fie un *cluster* alocabil în *FAT12* și *FAT16* dar este posibil ca marcajul `0x0FFFFFF7` să fie numărul unui *cluster* alocabil pe volumele *FAT32*.

Pentru a evita posibila confuzie a utilitarelor de disc, nici un volum *FAT32* nu ar trebui niciodată configurat astfel încât valoarea `0x0FFFFFF7` să reprezinte numărul unui *cluster* alocabil.

Lista *cluster*-elor libere din *FAT* nu este nimic altceva decât lista tuturor *cluster*-elor care conțin valoarea zero în intrarea lor din *FAT*. Această valoare trebuie prelucrată după cum s-a explicat anterior; pentru orice altă valoare, intrarea în corespunzătoare *FAT* nu este liberă.

Acestă listă cu *cluster*-ele libere nu este stocată pe disc; trebuie să fie calculată când volumul este montat, prin scanarea *FAT*-ului pentru intrări care conțin valoarea 0. Pe volumele *FAT32*, sectorul `BPB_FSInfo` poate conține un număr valid de *cluster*-e libere ale volumului (mai multe informații despre sectorul `FSInfo` din *FAT32* veți găsi într-un articol viitor).

Acum să vedem ce reprezintă cele două *cluster*-e rezervate din *FAT* (*cluster*-ele 0 și 1).

Primul *cluster* rezervat, `FAT[0]` conține valoarea `BPB_Media` pe cei mai puțin semnificativi opt biți și restul biților au valoarea 1.

De exemplu, dacă `BPB_Media` are valoarea `0xF8`, pentru *FAT12* avem `FAT[0]=0x0FF8`, pentru *FAT16* avem `FAT[0]=0xFFF8` și pentru *FAT32* avem `FAT[0]=0x0FFFFFF8`.

Al doilea *cluster* rezervat, `FAT[1]`, este setat la formatare cu valoarea *EOC*.

În volumele *FAT12*, acest *cluster* nu este folosit și conține întotdeauna valoarea *EOC*.

Pentru *FAT16* și *FAT32*, *driver*-ul sistemului de fișiere poate folosi cei mai semnificativi 2 biți din intrarea `FAT[1]` ca *flag*-uri de volum (partiție) care indică faptul că volumul (partiția) nu a fost "demontat" înainte de oprirea sistemului (ceilalți biți au întotdeauna valoarea 1).

Pentru *FAT16* avem `ClnShutBitMask=0x8000` și `HrdErrBitMask=0x4000`.

Pentru *FAT32* avem `ClnShutBitMask=0x08000000` și `HrdErrBitMask=0x04000000`.

Dacă valoarea `ClnShutBitMask&FAT[1]` este 1, înseamnă că volumul este "curat".

Dacă acest bit are valoarea este 0, înseamnă că *driver*-ul sistemului de fișiere nu a "demontat" volumul corect, după ultima utlizare.

Dacă valoarea `HrdErrBitMask&FAT[1]` este 1 înseamnă că nu s-au semnalat erori de citire/scriere în decursul sesiunii de lucru anterioare.

Dacă acest bit are valoarea 0, înseamnă că *driver*-ul sistemului de fișiere a întâlnit erori de citire/scriere pe volum în timpul ultimei utilizări, ceea ce indică posibilitatea existenței unor sectoare "*bad*" pe volumul respectiv.

Mai sunt de adăugat încă două lucruri importante despre zona de *FAT* a unui volum:

1. Ultimul sector din *FAT* nu este neapărat folosit în totalitate de către *FAT*. *FAT*-ul se oprește la *cluster*-ul care corespunde intrării cu numărul `CountofClusters+1` (a se vedea modul de calcul al valorii `CountofClusters` prezentat anterior), și această intrare nu este neapărat la sfârșitul ultimului sector din *FAT*. Codul de *FAT* nu ar trebui să facă nici un fel de presupuneri care se referă la conținutul ultimului sector de *FAT*, aflat după intrarea `CountofClusters+1`. La formatarea volumului toți biții situați după această intrare trebuie să fie inițializați cu valoarea 0.
2. Valoarea `BPB_FATSz16` (respectiv `BPB_FATSz32` pentru *FAT32*) poate fi mai mare decât este necesar. Cu alte cuvinte, la sfârșitul *FAT*-urilor pot exista sectoare neutilizate. Din acest motiv ultimul sector folosit al *FAT*-ului este întotdeauna calculat folosindu-se valoarea `CountofCluster+1` și niciodată folosind valoarea `BPB_FATSz16/32`. Codul de *FAT* nu ar trebui să facă nici un fel de presupuneri referitoare la conținutul acestor sectoare suplimentare din *FAT*. La formatarea volumului, aceste sectoare suplimentare ar trebui inițializate cu valoarea 0.

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.