Sistemes de fitxers FAT16 i FAT32

Descripció i aplicació

Documents Treball de Recerca
Gerard Marull Paretas
Abril 2007

Índex

1. Introducció

Com s'havia descrit anteriorment, el reproductor necessita interpretar les dades rebudes a través de la targeta SD. La informació pot organitzar-se de moltes maneres, des de la més simple com per exemple dades encadenades l'una darrera l'altra, fins a complexos sistemes de fitxers on es desen grans quantitats de dades, que a nivell d'usuari poden oferir la creació de fitxers, l'organització en carpetes, l'addició d'atributs i permisos, etc. El nostre reproductor, serà capaç d'interpretar dos formats: FAT16 i FAT32.

FAT (File Allocation Table), un sistema de fitxers desenvolupat per Microsoft, ha presentat diverses versions des del seu inici. El nombre que l'acompanya indica el nombre de bits utilitzats per desar les adreces dels clústers, cosa que implicarà en el tamany màxim de la partició. Són dos formats relativament simples, amb compatibilitat gairebé en tots els ordinadors i que són molt semblants entre ells. FAT32 no és res més que una millora de FAT16, corregint-ne alguns aspectes funcionals i millorant-ne d'altres com el tamany màxim de la unitat. Ja que en l'actualitat el tamany d'una targeta SD pot arribar fins als 8GB i FAT16 no suporta un tamany superior a 2GB, s'ha decidit implementar els dos sistemes.

El material necessari és el mateix descrit en l'anterior capítol, afegint el següent:

- Editor hexadecimal (En aquest cas KHexEdit)
- Lector de targetes SD per ordinador
- Imatge en un fitxer d'una la targeta SD
- Calculadora per a programadors per realitzar canvis de base, i càlculs a nivell de bits (En aquest cas Clarence)

2. Anàlisi del sistema de fitxers FAT

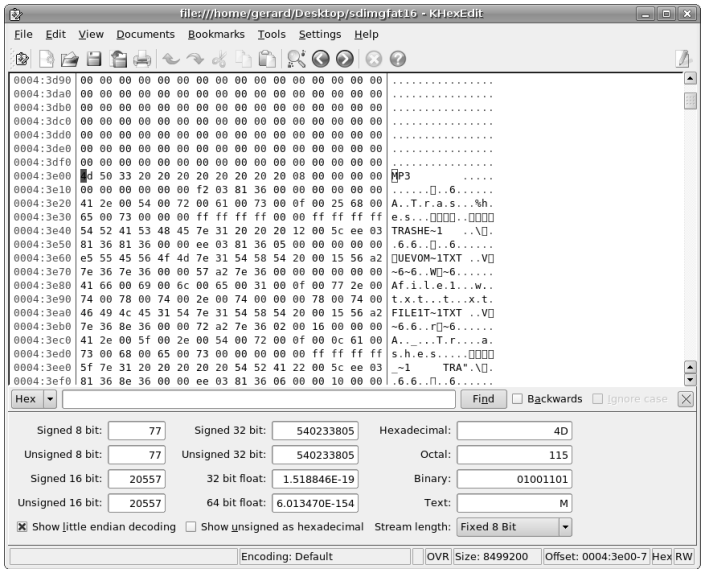
Seguidament s'explicarà en detall i d'una manera comprensiva com funciona el sistema de fitxers FAT i com interpretar-lo a través del microcontrolador. S'aconsella la utilització d'un sistema operatiu UNIX (Linux, MacOS X, etc.) per poder dur a terme tots els passos descrits a continuació.

2.1. Preparació

Abans de començar a introduir-nos en la interpretació, es recomana la creació d'un fitxer que contingui exactament el mateix que la targeta SD. Per fer-ho, disposem d'una manera molt fàcil sota qualsevol sistema UNIX, sols hem d'executar la següent comanda:

```
$ dd if=/dev/sdx of=/ruta/del/fitxer
```

On al paràmetre *if* hem d'introduir el dispositiu on tenim la targeta SD connectada i a *of* la ruta on volem desar aquest fitxer. Es recomana utilitzar un altre paràmetre: el *count*. D'aquesta manera podem introduir quants sectors volem copiar (ex. *count=20000*) i així evitem crear fitxers de grans dimensions, ja que per fer proves i veure exemples no ens cal tota la memòria de la targeta.
Un cop creada, podrem obrir aquest fitxer amb un editor hexadecimal. Això ens permetrà visualitzar els valors que conté la targeta SD en qualsevol posició de la memòria. És un mètode ràpid i eficaç de comprovar que les dades que el microcontrolador retorna siguin les correctes, com també el fet de poder navegar per tot el sistema de fitxers analitzant cadascuna de les parts.



- *Imatge de l'editor utilitzat mostrant part del contingut d'una targeta SD*

2.2. MBR

Ja que un disc qualsevol pot contenir diverses particions, haurem d'obtenir la localització de la primera. La resta, no s'interpretaran per part del reproductor ja que en targetes de tamany petit, és absurd desar la música de manera separada i també és un malbaratament innecessari de recursos per part del microcontrolador. Les dades d'aquesta primera partició, es troben concretament entre les posicions 0x1be i 0x1cd del primer sector (també anomenat MBR –

Master Boot Record).
En la següent taula podem observar com s'estructura l'MBR:

Posició	Descripció	Tamany
0x000	Codi executable (Sols utilitzat en els ordinadors)	446 bytes
0x1be	1a entrada de partició	16 bytes
0x1ce	2a entrada de partició	16 bytes
0x1de	3a entrada de partició	16 bytes
0x1ee	4a entrada de partició	16 bytes
0x1fe	Marcador d'executable (0x55 i 0xaa)	2 bytes

Així doncs, ens haurem de centrar just en els 16 bytes de l'entrada de la primera partició. Cada un dels bytes o conjunt de bytes conté un valor relacionat amb la partició. La següent taula ens mostra l'estructura d'aquests 16 bytes:

Posició	Descripció	Tamany
0x00	Estat de la partició (0x00 = Inactiva, 0x80 = Activa)	1 byte
0x01	Començament de la partició (Cara)	1 byte
0x02	Començament de la partició (Cilindre / Sector)	2 bytes
0x04	Format de la partició	1 byte
0x05	Fi de la partició (Cara)	1 byte
0x06	Fi de la partició (Cilindre / Sector)	2 bytes
0x08	Distància entre l'MBR i el primer sector de la partició	4 bytes
0x0c	Nombre de sectors en la partició	4 bytes

D'aquesta taula sols ens interessaran alguns dels valors. El primer serà el format de la partició. Aquesta posició conté un valor diferent per cada tipus de format. Perquè una partició sigui FAT16 haurà de contenir el valor 0x06 i per FAT32 haurà de contenir 0x0b o bé 0x0c. D'aquesta manera, podrem avisar amb un senyal d'error si la targeta que estem utilitzant conté algun altre format no suportat. La segona posició imprescindible és la 0x08. Aquesta ens permetrà saber en quin sector comença la partició. La resta de posicions no són utilitzades en aquest mòdul.

2.3. Estructuració dels formats FAT16 i FAT32

A partir del primer sector de la partició, trobarem les següents parts:

FAT16	FAT32
Registre d'arrancada FAT16	Registre d'arrancada FAT32
Taules FAT	Taules FAT
Entrades del directori arrel	Entrades de directori i fitxers
Entrades de directori i fitxers	

Tot seguit, s'explicarà cadascuna d'elles en detall.

2.3.1. Registre d'arrancada FAT16 i FAT32

Tot i que el seu contingut no és exactament igual en els dos formats, en totes dues trobarem informació essencial sobre la partició, com per exemple el sector/clúster[1] on comencen les dades, el tamany dels clústers, etc.

Estructura del Registre d'arrancada FAT16:

Posició	Descripció	Tamany
0x00	Codi màquina de salt (JUMP + NOP)	3 bytes
0x03	Nom OEM (Fabricant)	8 bytes
0x0b	Bytes per sector	2 bytes
0x0d	Sectors per clúster	1 byte
0x0e	Sectors Reservats	2 bytes
0x10	Nombre de còpies FAT	1 byte
0x11	Nombre màxim d'entrades a l'arrel	2 bytes
0x13	Nombre de sectors (Part. < 32Mb)	2 bytes
0x15	Descriptor del medi (suport)	1 byte
0x16	Sectors per FAT	2 bytes
0x18	Sectors per pista	2 bytes
0x1a	Nombre de cares	2 bytes
0x1c	Sectors ocults	4 bytes
0x20	Nombre total de sectors	4 bytes
0x24	Número de la unitat	2 bytes
0x26	Signatura (0x29)	1 byte
0x27	Número de sèrie de la partició	4 bytes
0x2b	Etiqueta de la partició	11 bytes
0x36	Nom del FAT (FAT16)	8 bytes
0x3e	Codi màquina executable	448 bytes
0x1fe	Marcador d'executable (0x55 i 0xaa)	2 bytes

- Els valors que s'utilitzaran d'aquesta taula són:

  – Nombre màxim d'entrades a l'arrel (0x11)
  – Nombre de còpies FAT (0x10)
  – Sectors per clúster (0x0d)
  – Sectors reservats (0x0e)
  – Sectors per FAT (0x16)

1 - Vegeu el glossari

Estructura del Registre d'arrancada FAT32:

Posició	Descripció	Tamany
0x00	Codi màquina de salt (JUMP + NOP)	3 bytes
0x03	Nom OEM (Fabricant)	8 bytes
0x0b	Bytes per sector	2 bytes
0x0d	Sectors per clúster	1 byte
0x0e	Sectors reservats	2 bytes
0x10	Nombre de còpies FAT	1 byte
0x11	- No utilitzat -	2 bytes
0x13	- No utilitzat -	2 bytes
0x15	Descriptor del medi (suport)	1 byte
0x16	- No utilitzat -	2 bytes
0x18	Sectors per pista	2 bytes
0x1a	Nombre de cares	2 bytes
0x1c	Sectors ocults	4 bytes
0x20	Nombre total de sectors	4 bytes
0x24	Nombre de sectors per FAT	4 bytes
0x28	Bits 0-4: Indiquen quina és la còpia de la Taula FAT activa. Bit 7: Indiquen si la sincronització de les Taules FAT es troba activa (FAT Mirroring).	2 bytes
0x2a	Versió del FAT32	2 bytes
0x2c	Clúster on comença el directori arrel	4 bytes
0x30	Sector d'informació del sistema de fitxers	2 bytes
0x32	Sector de la còpia de seguretat del sector d'arrancada	2 bytes
0x34	Reservat	12 bytes
0x40	Número de la unitat	1 byte
0x41	- No utilitzat -	1 byte
0x42	Signatura (0x29)	1 byte
0x43	Número de sèrie	4 bytes
0x47	Etiqueta de la partició	11 bytes
0x52	Nom del FAT (FAT32)	8 bytes
0x5a	Codi màquina executable	420 bytes
0x1fe	Marcador d'executable (0x55 i 0xaa)	2 bytes

- Els valors significatius que s'utilitzen d'aquesta partició són:

- Clúster on comença el directori arrel (0x2c) - *normalment 2*
- Nombre de còpies FAT (0x10)
- Sectors per clúster (0x0d)
- Sectors reservats (0x0e)
- Sectors per FAT (0x24)

2.3.2. Taules FAT

Zona on es troba aquesta part (Valors descrits en els punts 2.2 i 2.3.1):

- *Primer sector de la partició + Sectors reservats*

El sistema de fitxers FAT, s'organitza en blocs anomenats clústers de tamany variable (de 2kb a 32kb, depenent del tamany de la unitat). Un fitxer pot ocupar un o més clústers en funció del seu tamany. Els clústers d'un fitxer es poden trobar en diverses parts del disc (no necessàriament consecutives) cosa que en provoca la seva segmentació. Aquesta taula, ens servirà per obtenir l'ordre en què es van succeint. En cada entrada trobarem el següent clúster on continua el fitxer o llistat d'entrades de directori. FAT16 s'organitza en entrades de 2 bytes, i FAT32 en entrades de 4 bytes. També poden contenir valors per indicar el final del fitxer (0xffff per FAT16 i 0x0fffffff per FAT32), clúster defectuós, clúster lliure i clúster reservat. També cal dir que sol existir més d'una còpia d'aquesta taula, ja que la seva pèrdua faria pràcticament irrecuperable les dades contingudes en la partició.

Exemple: Suposem que un determinat fitxer comença en el clúster 3. Comencem a llegir-lo, fins que arribem al final del clúster actual. Per saber on continua, ens dirigim a la 3a entrada de la Taula FAT, i llegim el valor 7. Això voldrà dir que haurem de continuar llegint en el clúster 7, i així successivament fins que trobéssim el valor indicador de final de fitxer.

2.3.3. Entrades del directori arrel

Zona on es troba aquesta part (Valors descrits en els punts 2.2 i 2.3.1):

- *Primer sector de la partició + Sectors reservats + (Nombre de còpies FAT * Sectors per FAT)*

Aquesta part és exclusiva del FAT16. Ja que Microsoft ha tingut una vella tradició en fer les coses mal fetes, resulta que sota FAT16 les entrades del directori arrel (el llistat de fitxers que conté una carpeta) es tracten per separat. Així que aquest directori té un nombre limitat de fitxers (descrit en el Registre d'arrancada FAT16) que sol ser de 512. L'organització d'aquesta part es descriurà tot seguit.

2.3.4. Entrades de directori i fitxers (Dades)

Zona on es troba aquesta part (Valors descrits en els punts 2.2 i 2.3.1):

- ***FAT16*** → *Primer sector de la partició + Sectors reservats + (Nombre de còpies FAT * Sectors per FAT) + ((Nombre màxim d'entrades a l'arrel * 32) / Bytes per sector)*
- ***FAT32*** → *Primer sector de la partició + Sectors reservats + (Nombre de còpies FAT * Sectors per FAT)*

A partir d'aquí ens trobarem la resta d'entrades de directori i continguts de fitxers, és a dir, totes les dades desades en el disc. Sota FAT32 el directori arrel comença aquí i per tant es tracta com la resta i la limitació del nombre de fitxers desapareix.
L'estructuració en detall d'aquest apartat i de l'anterior, es farà en el següent punt ja que consta de diversa complexitat.

2.4. Entrades de directori i LFN

Començant en el directori arrel, trobarem tot el llistat de fitxers que conté aquest directori. Cada entrada té un tamany de 32 bytes i ens dona informació sobre un fitxer determinat, com per exemple el nom, data de creació, en quin clúster comença el contingut del fitxer, etc. Vegem doncs com s'estructuren aquestes entrades

Posició	Descripció	Tamany
0x00	Nom Valors especials: 0x00: L'entrada es troba buida, indicant que ja no hi ha més fitxers en el directori. 0xe5: Indica que aquest fitxer ha estat eliminat 0x2e: Entrades punt (“.” i “..”) - Clúster actual i clúster de l'arrel	8 bytes
0x08	Extensió	3 bytes
0x0b	Atributs: Bit 0: Només de lectura Bit 1: Ocult Bit 2: Sistema Bit 3: Etiqueta de volum Bit 4: Directori Bit 5: Fitx. de memòria cau (Cache) Bit 6-7: No utilitzats - Un atribut es troba assignat si el valor del seu bit és 1.	1 byte
0x0c	Reservat	1 byte
0x0d	Hora de creació (Unitats ms, 0-199)	1 byte
0x0e	Hora de creació (Hora, minuts i seg.) Bits 0-4: Segons / 2 (0-29) Bits 5-10: Minuts (0-59) Bits 11-15: Hora (0-23)	2 bytes
0x10	Data de creació Bits 0-4: Dia (1-31) Bits 5-8: Mes (1-12) Bits 9-15: Any (0=1980, 127=2107)	2 bytes
0x12	Última data d'accés (Estructura de 0x10)	2 bytes
0x14	FAT12 i FAT16: Índex EA (OS/2 i NT) FAT32: 2 bytes superiors del clúster	2 bytes
0x16	Última hora de modificació (Estructura de 0x0e)	2 bytes
0x18	Última data de modificació (Estructura de 0x10)	2 bytes
0x1a	FAT12 i FAT16: Primer clúster del fitxer FAT32: 2 bytes inferiors del primersclúster (Vegeu 0x14)	2 bytes
0x1c	Tamany del fitxer (en bytes)	4 bytes

Aquesta estructura però, té una limitació força important. Si ens fixem en el camp "Nom", veurem que la seva llargada màxima és de 8bytes, cosa que ens limita la llargada del nom a 8 caràcters ASCII[2]. Aquest tipus de noms eren utilitzats en els ordinadors antics, i presentaven un aspecte com el següent:

- FILE1T~1.TXT
- FILE2T~1.DOC

En la informàtica actual és un sistema obsolet, ja que la seva llargada és massa curta, i no suporta caràcters especials com *ç* i *ñ* o d'un altre alfabet, accents, etc. Així doncs, Microsoft va fer un "gran invent" creant els LFN (*Long File Names*), que suporten una llargada de fins 255 caràcters, i canvia el sistema de caràcters a UTF-16. Per fer-ho va decidir aplicar la tècnica del pedaç, aprofitant el sistema de fitxers sense haver-lo de modificar i fent també que fos compatible amb màquines velles que no suportaven LFN. Seguidament, descriurem com s'organitzen els LFN:

Estructura d'una entrada LFN:

Posició	Descripció	Tamany
0x00	Camp ordinal	1 byte
0x01	Caràcter UTF-16 1	2 bytes
0x03	Caràcter UTF-16 2	2 bytes
0x05	Caràcter UTF-16 3	2 bytes
0x07	Caràcter UTF-16 4	2 bytes
0x09	Caràcter UTF-16 5	2 bytes
0x0b	Atribut	1 byte
0x0c	Tipus (Sempre 0)	1 byte
0x0d	Checksum	1 byte
0x0e	Caràcter UTF-16 6	2 bytes
0x10	Caràcter UTF-16 7	2 bytes
0x12	Caràcter UTF-16 8	2 bytes
0x14	Caràcter UTF-16 9	2 bytes
0x16	Caràcter UTF-16 10	2 bytes
0x18	Caràcter UTF-16 11	2 bytes
0x1a	Clúster (No utilitzat, sempre 0)	2 bytes
0x1c	Caràcter UTF-16 12	2 bytes
0x1e	Caràcter UTF-16 13	2 bytes

Les entrades LFN comparteixen espai amb les entrades normals, per això tenen el mateix tamany. Així que necessitarem algun mètode poder-les identificar de la resta. A més, hem de tenir en compte que els sistemes vells tracten totes les entrades com a normals, i per tant, hem de trobar una manera de què siguin ignorades. Si posem els primers quatre atributs de fitxer a 1 (0b00001111 ó 0xf), obtenim un fitxer de tipus:

- Només de lectura

2 Vegeu document Annex *"Sistemes de caràcters en l'informàtica: ASCII i UTF-16"*

- Ocult
- Fitxer de sistema
- Etiqueta de volum

Això farà que sigui un fitxer ignorat per la majoria de programes, i per tant evitarà problemes en màquines sense suport perLFN.
Tal com podem observar a la taula, en cada entrada de directori només podem desar 13 caràcters i per tant necessitarem més d'una entrada per a fitxers que superin aquesta llargada. Conseqüentment, ens quedaran diverses entrades que juntament contindran el nom total. Per tal de poder generar el nom complet, necessitarem algun valor que ens indiqui l'ordre en què es troben llistades les diferents entrades. La solució es troba en el primer camp (Camp ordinal), ja que conté aquest número. Finalment, sols ens caldrà un altre identificador per detectar l'última entrada. Si trobem el 6è bit a 1 de la primera posició d'una entrada a més del número, indicarà l'últim camp del conjunt.

3. Aplicació en el microcontrolador

Finalment ens queda aplicar el que hem explicat programant el microcontrolador i provant-lo en un circuit. Seguidament es farà una breu explicació de com funciona el programa, és a dir, quins passos duu a terme per executar la tasca amb èxit.
Aquest apartat es pot seguir juntament amb el codi que es troba en els fitxers *main.c*, *fat.c* i *fat.h* de la carpeta Codi del CD-ROM, tot i que no és necessari si no es desitja entrar a la part tècnica.

NOTA: La versió d'aquest mòdul no és definitiva. Algunes funcions hi manquen i també pot contenir errors.

3.1. Definicions, estructures i variables

El fitxer *fat.h*, conté una sèrie de definicions, estructures de dades, variables de tipus extern, etc. per ser utilitzades des de qualsevol part del programa que inclogui aquest fitxer com a capçalera. A més, inclou els prototips de les funcions de *fat.c*, així permetent-ne l'accés des d'altres mòduls del programa.

3.2. Funcions generals

En el mòdul FAT hi han dues funcions que són utilitzades força sovint per la resta de funcions. Es tracta de *fat_getnextcluster()* i *fat_clustertosector()*. La primera, és l'encarregada d'obtenir el següent clúster a partir del que l'hi és donat (Vegeu punt 2.3.2), i la segona converteix un clúster donat al seu corresponent sector. El concepte de clúster sols és utilitzat per FAT, la SD treballa a partir de sectors. Per fer-ho segueix els següents passos:

1. Comprovació de si el clúster donat és 0. Si és 0 serà per força el sector arrel d'una partició FAT16. (Recordeu que FAT16 tracta el directori arrel per separat)
2. Per la resta de valors, podem aplicar la següent fórmula:
   *(Clúster donat - 2) * Sectors per clúster + Primer sector de dades*

3.3. Inicialització del sistema de fitxers

Aquesta part correspon a la funció *fat_init()* del fitxer *fat.c*. És el primer pas que haurem de dur a terme per poder navegar a través de les dades que conté la partició. Aquesta funció segueix els següents passos:

1. Lectura del primer sector de la targeta SD (0).
2. Comprovació del sistema de fitxers (Només continuarà si es troba FAT16 o FAT32)
3. Obtenció de les dades de la 1a partició. (Vegeu punt 2.2)
4. Lectura del sector del Registre d'arrancada FAT16/FAT32. (Vegeu punt 2.3.1)
5. Obtenció de les dades necessàries sobre la partició.
6. Finalització amb èxit. Retorna 1[3].

3.4. Obtenció del nombre de fitxers dins d'una carpeta

Aquesta part correspon a la funció *fat_getentries()* del fitxer *fat.c*. Requereix dos arguments: *cluster* i *format*. A *cluster* hem d'introduir el clúster on es troba el començament de la carpeta

3 En les nostres funcions, un valor de retorn 1 significarà que s'ha processat amb èxit, i 0 si s'ha esdevingut algun error. Altres valors poden correspondre a codis d'error específics.

que volem llegir. Això ens permet navegar per qualsevol subcarpeta i no només pel directori arrel. El segon paràmetre, *format*, hi podem introduir un format específic (ex. "MP3"), i només llistarà fitxers en aquell format. Si volem obtenir tot el contingut de la carpeta, hem de passar un punter NULL. Procediment que segueix aquesta funció:

1. Entrada en un bucle infinit, que serà l'encarregat de navegar recursivament un directori fins al final.
2. Comprovem si hem arribat al final del sector actual. Hi haurà un màxim de 16 entrades en un sector, ja que el tamany de sector serà sempre de 512 bytes i si recordem el tamany de cada entrada (32bytes) tindrem que 512 / 32 = 16. Si és afirmatiu, procedirem a comprovar si hem arribat al final del clúster actual. Si és així, executarem *fat_getnextcluster()* i continuarem. Finalment, llegirem el següent sector de la targeta SD.
3. Un cop passat el pas anterior, comprovarem si existeixen més fitxers a la carpeta mirant si la primera posició de l'entrada actual val 0. Si es així sortirem del bucle i es retornarà el nombre de fitxers acumulat.
4. Ja que a partir d'ara només ens interessa el nombre de fitxers, només haurem de comptar aquelles entrades que no siguin fitxers eliminats ni tampoc entrades LFN (Vegeu punt 2.4). Sí és així, només ens caldrà comprovar si hem demanat a la funció algun format en concret o bé tots els fitxers del directori i incrementarsi s'escau.

3.5. Obtenció de la informació sobre un fitxer

Aquest apartat correspon a la funció *fat_getentryinfo()* del fitxer *fat.c*. La utilitzarem per obtenir dades importants sobre un fitxer (tamany, localització, etc.). Requereix quatre paràmetres. Al primer (*cluster*) li haurem de passar el clúster del directori on cercar el fitxer. El segon (*entrynumber*) i tercer (*filename*) paràmetre, s'emplenaran segons com volem obtenir les dades del fitxer:

- A partir de l'ordre en que es troba localitzada en el directori: Haurem de passar aquest nombre al segon paràmetre, i un punter NULL al tercer.
- A partir del seu nom: Haurem de passar el valor 0 al segon paràmetre, i el nom del fitxer al tercer.

A l'últim (*file*) li haurem de passar una estructura ja creada basada en el model per dades d'un fitxer (*struct fat_fileentry* a *fat.h*). Procediment que segueix aquesta funció:

1. Repetició dels passos 1 i 2 del punt 3.4.
2. Comprovem si hem arribat a final de la carpeta, comprovant si el primer valor de l'entrada val 0. Si és així sortirem del bucle. Nota: La sortida del bucle implica el retorn del valor 0 per part de la funció, això voldrà dir que no s'ha trobat un fitxer amb les característiques demanades.
3. Comprovem que no ens trobem davant una entrada eliminada, i si és així, tindrem les següents opcions:
   - És una entrada LFN: Entrarem a desar-ne les dades si hem passat un nom a la funció, sinó seria absurd. (Vegeu punt 2.4)
   - És una entrada de fitxer normal: (S'ignoraran arxius ocults, etiquetes de volums i arxius de sistema). Aquí incrementarem el nombre d'entrades. Comprovarem si hem estat desant un LFN anteriorment, i sinó desarem l'altre nom del sistema vell, ja que es podran utilitzar tots dos tipus de noms per obtenir informació d'un fitxer.
4. Finalment, mirarem si les dades obtingudes fins ara compleixen les condicions demanades (nom o número d'entrada), i si és així desarem les dades següents del fitxer:

- Primer clúster (Variarà el mètode d'obtenció segons sigui FAT16 o FAT32)
- Tipus (Fitxer o carpeta)
- Tamany

5. Retorn del valor 1 per part de la funció.

3.6. Lectura del contingut d'un fitxer

Aquest apartat correspon a la funció *fat_readfile()* del fitxer *fat.c*. Requereix dos arguments, el primer (*file*) una estructura ja creada i anteriorment emplenada amb la informació d'un fitxer i el segon (*length*) la llargada de de bytes que en volem llegir (Màx. de 512). Procediment que segueix aquesta funció:

1. Comprovació que el tamany del fitxer no sigui 0 (fitxer buit), i que el clúster actual tampoc sigui 0 (fitxer buit o ha estat acabat de llegir).
2. Comprovació que el la suma del valor que desa l'última posició d'accés al fitxer i la llargada que hem donat no sobrepassin el final del fitxer. Si es donés el cas, només es llegiria fins a final de fitxer, però no més enllà.
3. Càlcul del sector que hem de llegir, passant l'actual clúster del fitxer a sector, i sumant aquells que hem avançat dins del clúster actual.
4. Lectura del sector obtingut, i suma de la llargada llegida al valor que desa l'última posició d'accés.
5. Comprovació que no hem arribat a final de clúster (El valor 0 en els sectors avançats implica l'obtenció del següent clúster – vegeu la funció en el codi per més detalls).
6. Retorn del total de bytes llegits.

3.7. Comprovació del funcionament

Fnalment, ens queda comprovar que el codi programat funciona. Per fer-ho, escriurem un petit codi al a la funció *main()* (primera funció executada pel microcontrolador) del fitxer *main.c*. Que executi algunes d'aquestes funcions. Abans de començar, haurem de desar un fitxer a la targeta SD que ens servirà per la comprovació. El tamany del fitxer ha de ser petit (ex. de 100kb) ja que sinó tardaria molt en enviar-nos les dades. Un cop fet, només cal obrir el programa que mostra les dades rebudes a través del port sèrie (en aquest cas Cutecom) i connectar el circuit. El procediment que segueix aquesta funció és el següent:

1. Inicialització del sistema de fitxers (*fat_init()*).
2. Obtenció del nombre de fitxers en el directori arrel mitjançant la funció *fat_getentries()* i enviament a través del port sèrie en format decimal.
3. Obtenció de la informació sobre el fitxer que hem desat anteriorment utilitzant el seu nom i enviament de les dade obtingudes.
4. Entrada en un bucle, encarregat de demanar les dades del fitxer que hem creat fins que es retorni que la llargada llegida és 0 (fi del fitxer).
5. En cada lectura, s'enviaran les dades llegides en format hexadecimal i alineades de la mateixa manera que l'editor hexadecimal per poder fer la comparació més fàcilment.
6. Contrastació de les dades: Podrem comprovar que les dades rebudes corresponen a la nostra partició, i pel que fa al fitxer, podem obrir-lo a l'editor hexadecimal, i comprovar que el conjunt de dades coincideix des del principi fins al final.

4. Bibliografia

BANNACK, Angelo i BRUNO, Giordano. "FAT16/32 Driver for ATMEL AVR". http://www.e-armazem.com.br/dev/fat16-32driver/

DOBIASH, Jack. "FAT16 Structure Information", http://home.teleport.com/~brainy/fat16.htm

DOBIASH, Jack. "FAT32 Structure Information", http://home.teleport.com/~brainy/fat32.htm

EVANS, Allan. "FAT16 Interface for MSP430", Michigan State University, 2004.

SCHILDT, Herbert. *Programación en C, Cuarta edición*. Madrid: McGraw Hill, 2001.

http://en.wikipedia.org/wiki/File_Allocation_Table

http://home.teleport.com/~brainy/lfn.htm

http://www.atmel.com/avr

http://www.avrfreaks.net