SOA

CopyCat

Manel Roca Relats (ls27599)

Índex

Requisits

FASE 1

La primera part de la pràctica consisteix en crear una aplicació que extregui la informació de dos tipus de sistemes de fitxers, FAT16 i EXT2.

Per tal de realitzar aquesta informació es disposa de la documentació oficial de cadascuna de les extensions per tal de saber on es troba la diferent informació necessària dins el fitxer binari.

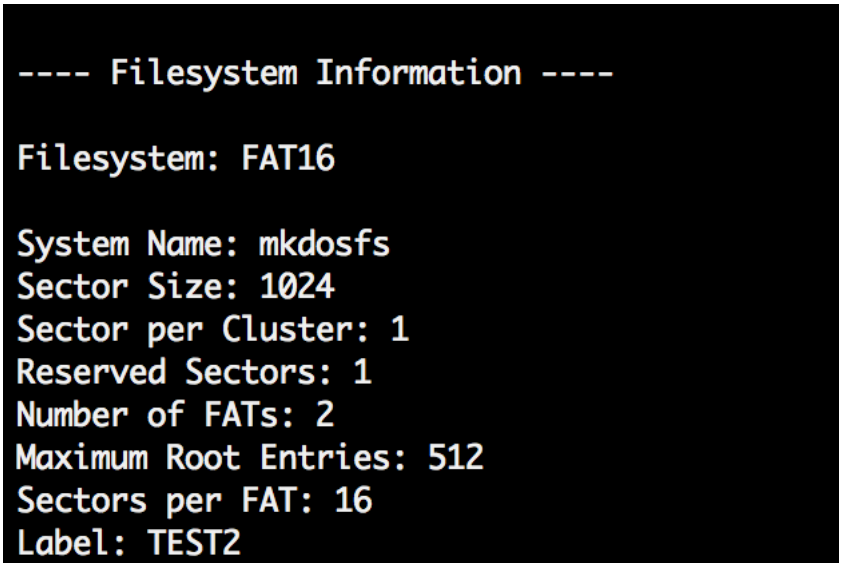
L'aplicació rep dos paràmetres per línia de comandes. La primera d'elles és l'opció a executar, en aquesta fase la opció és "/info". El segon parametre és el volum que s'ha d'analitzar.

Per l'extensió EXT2 s'ha de guardar:

```
---- Filesystem Information ----

Filesystem: EXT2

INODE INFO
Inode Size: 128
Number of inodes: 1024
First Inode: 11
Inode Group: 1024
Free inodes: 968

BLOCK INFO
Block size: 1024
Reserved Blocks: 204
Free Blocks: 966
Total Blocks: 4092
First Block: 1
Block Group: 8192
Block Group: 8192

VOLUME INFO
Volume name: TEST1
Last check: Sat Apr  2 14:31:02 2011
Last mount: Sat Apr  2 14:42:36 2011
Last written: Sat Apr  2 14:57:31 2011
```

Els volums amb l'extensió EXT2 guarden la informació que es necessita per aquesta fase en un sector anomenat "superblock". Aquest espai de memòria està situat en el byte 1024 i sempre ocupa 1K de memòria. Dins aquest block d'1K de memòria és on troba tota la informació referent al volum que s'està processant.

En canvi per l'extensió FAT16 s'ha de guardar:



Als volums amb extensió FAT16 la informació sempre està als primers bytes del volum, i aquesta és de mida constant.

Finalment s'ha de mostrar per pantalla com es mostra en els exemples anteriors.

FASE 2

La segona part de la pràctica consisteix en utilitzar l'aplicació de la fase 1 per tal de buscar fitxers dins del volum proporcionat.

Per línia de comandes s'introdueixen tres paràmetres a part del nom de l'executable. Els paràmetres són en primer lloc el nom de la funció que es vol realitzar. En aquest cas és "/find". El segon paràmetre és el nom del volum a buscar. Finalment l'últim paràmetre és nom del fitxer a buscar.

Per poder realitzar aquesta funcionalitat s'ha dividit el programa en dues parts. La primera part analitza el volum en el cas que aquest sigui FAT16. La segona part del codi és la que analitza el volum amb format EXT2.

FASE 3

A la tercera part de la pràctica es va demanar que s'implementés la comanda "cat" del fitxer introduït.

Per línia de comandes s'introdueixen tres paràmetres a part del nom de l'executable. Els paràmetres són en primer lloc el nom de la funció que es vol realitzar. En aquest cas és "/cat". El segon paràmetre és el nom del volum a buscar. Finalment l'últim paràmetre és nom del fitxer a buscar.

Per la pantalla s'ha de mostrar tot el contingut del fitxer.

FASE 4

La quarta fase de la pràctica es demanava implementar la funcionalitat de "/copy". Aquesta funcionalitat ha de copiar un fitxer d'un volum en format Ext2 a un volum en format FAT16.

Com a paràmetres tenim la comanda "/copy", el volum en format EXT2, el volum en format FAT16 i el nom del fitxer a copiar.

Disseny

FASE 1

Les dades del sistema FAT16 es troben en les següents posicions:

Nom	Offset (Bytes)	Mida (Bytes)
File System	510	2
System Name	3	8
Sector Size	11	2
Sector per Cluster	13	1
Reserved Sector	14	2
Number of FATs	16	1
Max root entries	17	2
Sectors per FAT	22	2
Label	43	11

En el camp File System s'ha posat la posició dels bytes que s'utilitzen per la comprovació del sistema de fitxers. Per sistemes FAT16 el valor d'aquests dos bytes ha de ser 0x55 i 0xAA, valor decimal de 43605.

Les dades del sistema EXT2 es troben a les següents posicions:

Nom	Offset (Bytes)	Mida (Offset)
File System	1024 + 56	2
Inode Size	1024 + 88	2
Number Inodes	1024 + 0	4
First Inode	1024 + 84	4
Inode Group	1024 + 40	4
Free Inodes	1024 + 16	4
Block Size [1]	1024 + 24	4
Reserved Blocks	1024 + 8	4
Free Blocks	1024 + 12	4
Total Blocks	1024 + 4	4
First Block	1024 + 20	4
Block Group	1024 + 32	4
Flag Group	1024 + 36	4
Volume Name	1024 + 120	16
Last Check [2]	1024 + 64	4
Last Mount [2]	1024 + 44	4
Last Written [2]	1024 + 48	4

[1] El valor de Block Size no és directament el valor del sistema de fitxers. S'ha de fer la següent operació per obtenir el valor desitjat: BlockSize = 1024 << block_size_llegit.

[2] El valor de les dates es llegeix com un valor enter, número de segons des del 01/01/1970. Per fer la transformació a la data que coneixem s'utilitza la funció ctime de la llibreria time.h

Com en el cas anterior, la informació de la fila de "File System" és els bytes que s'utilitzen per saber si és EXT2. El valor amb el que s'ha de comparar és 61267.

FASE 2

Per poder buscar un fitxer en un volum FAT16 primer s'ha de tractar el nom del fitxer. El nom del fitxer en FAT16 només pot tenir 8 caràcters, per tant si s'introdueix un nom de fitxer a buscar amb més de 8 caràcters s'ha de reduir. Quan un nom de fitxer supera els 8 caràcters els dos últims caràcters passen a ser ~M, on M és el número que identifica el fitxer, per si hi ha més d'un fitxer que comença igual.

Un cop tenim el nom del fitxer correcte s'obre el volum FAT16. En primer lloc ens saltem la "reserved region". Després ens saltem les copies de la FAT. En aquest moment ja estem a la regió de la "root directory" que conté 32B per entrada. De la "reserved region" hem obtingut el número màxim de entrades que hi ha.

El funcionament és un bulce que recorri totes les entrades de "root" o fins que trobi el fitxer que busquem. L'execució de cada iteració sempre és igual. Comencem llegint el nom del fitxer, avancem 10 bytes que és una zona reservada.

A continuació es llegeixen les dades necessàries per tractar el fitxer.

Finalment es comprova que sigui un fitxer, i si és així es comprova el nom.

Per buscar el fitxer en un volum EXT2 el que es fa és anar al inode 2 de la taula d'inodes (root directory). La informació d'on comença la taula d'inodes es troba 8 bytes després del superblock. Anem a on comença la taula d'inodes i avancem el fitxer el número de bytes que ocupa un inode. En aquest moment ja estem al inode 2, que és el de "root directory". D'aquest inode s'ha d'analitzar els diferents blocs associats. Primer es busca en els 12 punters directes de mida 4 bytes, que estan a 40 bytes de l'inici de l'inode. Cadascun d'aquests punters apunta a un bloc de dades que conté un conjunt de fitxers amb el nom del fitxer i el seu inode.

Després d'aquests 12 punters hi ha els 3 punters indirectes. El primer punter només té un nivell de profunditat, és a dir, aquest punter apunta a un bloc de dades que cada 4 bytes és un nou punter a dades. El segon punter té dos nivells, i el tercer tres.

Un cop s'ha trobat el nom del fitxer que es busca, es va fins l'inode del fitxer i es mostra la seva mida.

FASE 3

Per poder realitzar aquesta funcionalitat s'ha utilitzat la funcionalitat implementada anteriorment, és a dir, la comanda "/find".

En primer lloc es comprovava el tipus de volum que l'usuari proporcionava. En cas de no ser EXT2 es mostrava un missatge d'error.

Quan ja teníem que el volum era correcte es continuava buscant el nom de fitxer proporcionat per l'usuari dins del volum. Si el fitxer es trobava dins el volum, la funció de cerca retornava el número d'inode associat al fitxer, altrament retornava 0.

Un cop conegut el número d'inode del fitxer es desplaçava el volum fins al principi de l'inode corresponent. D'aquest inode es recuperava la informació de la mida del fitxer, per saber fins on pintar.

Finalment es recorrien els blocs associats als punters de l'inode i es mostrava la informació continguda dins del bloc per la pantalla fins haver escrit tants bytes com la mida del fitxer.

FASE 4

Per poder implementar aquesta fase s'ha utilitzat totes les funcionalitats dels apartats anteriors.

En primer lloc s'ha utilitzat les funcionalitats de "/info" per verificar i recuperar les dades necessàries dels dos volums.

Després s'ha utilitzat la opció de "/find" per verificar que el fitxer a copiar existia.

També s'ha utilitzat la funcionalitat de "/cat" per recuperar el contingut del fitxer.

Finalment s'ha implementat el "/copy" amb la informació obtinguda anteriorment.

Per realitzar aquesta funcionalitat s'ha buscat en primer lloc que el fitxer hi càpigues. Si el fitxer es podia guardar en el volum FAT el que es feia buscar clústers lliures i escriure el contingut en els clústers. Finalment s'enllaçava els diferents clústers i s'escrivia la informació en el root.

Tipus

FASE 1

Els tipus utilitzats en aquesta pràctica son dos, un per guardar es dades de FAT16 i un altre per guardar les dades de EXT2.

DadesFat

Per les dades de FAT16 tenim el següent tipus:

```
typedef struct{
    char sFileSystem[6];
    char sSystemName[20];
    unsigned short int snSectorSize;
    unsigned short int snSectorCluster;
    unsigned short int snReservedSectors;
    unsigned short int snNumFats;
    unsigned short int snMaxRootEntries;
    unsigned short int snSectorsPerFat;
    char sLabel[30];
}DadesFAT;
```

Aquest tipus serveix per guardar tota la informació processada del volum FAT16. La majoria de dades són del tipus `"unsigned short int"` ja que dins el volum FAT16 les dades numèriques es guarden en 2 bytes i en "little endian". Utilitzant el tipus `"unsigned short int"` el programa en c, al llegir les dades fa la conversió automàticament a un format que es pugui entendre com a valor.

DadesExt

Per les dades del sistema EXT2 s'ha utilitzat:

```
typedef struct{
    char sFileSystem[5];
    //Inode info
    unsigned short int snInodeSize;
    unsigned int nNumberInodes;
    unsigned int nFirstInode;
    unsigned int nInodeGroup;
    unsigned int nFreeInode;
    //Block info
    unsigned int nBlockSize;
    unsigned int nReservedBlocks;
    unsigned int nFreeBlocks;
    unsigned int nTotalBlocks;
    unsigned int nFirstBlocks;
    unsigned int nBlockGroup;
```

```
    unsigned int nFlagGroup;
    //Volume info
    char sVolumeName[20];
    unsigned int nLastCheck;
    unsigned int nLastMount;
    unsigned int nLastWritten;
}DadesEXT2;
```

El tipus DadesEXT2 serveix per emmagatzemar totes les dades referents al format EXT2. El tipus està dividit en tres blocs. El primer és la informació referent als "inodes" del volum. El segon bloc fa referència a la informació dels "blocks". Per últim, el tercer bloc guarda el nom del volum, i les dates de última modificació, última verificació i l'última vegada que el volum s'ha muntat.

FASE 2

```
typedef struct{
    char sName[9];
    char sExt[4];
    unsigned char cAttr;
    unsigned short int snTime;
    unsigned short int snDate;
    unsigned short int snStartCluster;
    unsigned int nFileSize;
}FATRootDirectory;

typedef struct{
    unsigned int numInode;
    unsigned short int registerLength;
    unsigned char nameLength;
    unsigned char fileType;
}DirectoryEXT2;
```

Els tipus mostrats anteriorment representen el contingut d'un directori dels diferents formats. S'utilitza per recuperar la informació del volum, tractar-la i poder guardar posteriorment la informació modificada en el volum.

FASE 3

Mateixos tipus utilitzats anteriorment.

FASE 4

Mateixos tipus utilitzats anteriorment.

Proves realitzades

FASE 1

Les proves realitzades en aquesta fase han sigut:

- Executar la pràctica amb els volums proporcionats.
- Verificar la informació de retorn amb els exemples de l'enunciat.

FASE 2

Les proves realitzades en aquesta fase han sigut:

- Muntar el volum en una raspberry.
- Seleccionar un fitxer i comprovar la seva mida.
- Executar la comanda buscant el fitxer en qüestió.
- Verificar la mida del fitxer amb el retorn de l'executable de la pràctica.

FASE 3

Les proves realitzades en aquesta fase han sigut:

- Descarregar els fitxers de l'eStudy.
- Executar la comanda amb el fitxer en qüestió redireccionant la comanda cap a un fitxer.
- Comprovar amb la comanda diff les diferencies entre fitxers.

FASE 4

Les proves realitzades en la fase 4 han sigut:

- Executar la comanda des del servidor "matagalls".
- Descarregar el volum resultant al ordinador personal.
- Pujar amb el FileZilla el volum a una raspberry.
- Muntar el volum
- Comprovar el contingut del fitxer copiat.

EXT2 a FAT16

La comanda de Linux que s'utilitza per tal de crear un volum FAT16 és "mkdosfas –F 16". Aquesta comanda té una sèrie de modificadors per tal de crear un volum de les mateixes característiques que el analitzat pel programa.
Els modificadors d'aquesta comanda són:

-C : especifica que la mida del cluster sigui la mínima per defecte.
-n : especifica el nom del volum "Label".
-S : mida del sector, normalment 512.
-s : número de clusters = Block Size/ mida sector (-S)

Problemes observats

FASE 1

El principal problema d'aquesta pràctica ha sigut poder obtenir la informació numèrica dels volums. Aquesta informació està guardada en "little endian" cosa que dificulta les coses. Utilitzant la comanda "hexdump" per poder comprovar els resultats, es va comprovar que aquesta comanda ja feia la conversió d'aquest format. En un primer moment, quan es desconeixia que aquesta eina feia la conversió, els resultats obtinguts no van ser els que es desitjaven i es va perdre temps fins que es va descobrir que no era un problema de codi, sinó que la comprovació que es realitzava no era correcta.

Finalment quan ja se sabia com utilitzar "hexdump" es va poder realitzar la pràctica sense cap altre problema.

FASE 2

Un dels problemes més importants que s'ha tingut i que ha suposat una major dedicació d'hores en la fase ha sigut els punters indirectes del sistema de fitxers EXT2. Aquest sistema va ser complex d'entendre i difícil de debugar. Per aquest motiu es va tardar més temps de l'esperat en finalitzar la fase.

Un altre problema que es va tenir va ser la posició dins del volum. Es va plantejar el problema de manera recursiva, però quan la funció recursiva acabava i tornava a la funció que l'havia cridat, el punter al volum estava desplaçat i no a la posició anterior a la crida. Per aquest motiu es tenia que guardar la posició inicial de cada crida que es feia per poder recuperar-la posteriorment. Es va solucionar aquest problema guardant el valor de retorn del lseek.

FASE 3

En aquesta fase s'han observat els mateixos problemes que la fase anterior, ja que el tractament de la informació és la única part que canvia.

Els problemes destacats són la posició del punter al volum, que a mesura que es va llegint informació varia la seva posició. I el tractament de la informació continguda dins els punters indirectes.

FASE 4

El principal problema que s'ha tingut en aquesta fase de la pràctica han sigut els dos primers clústers. El sistema FAT16 té dos clústers al principi que estan en memòria però que no s'utilitzen. A la hora de copiar el fitxer s'han de tenir en compte en les copies de la FAT, és a dir, a la hora de buscar un clúster lliure. Però a la hora de copiar la informació al clúster corresponent no s'ha de tenir en compte.

Un altre problema que s'ha tingut és el fet d'obrir el fitxer només en lectura. Quan s'intentava escriure el programa no generava cap missatge d'error, ja que no es controlava el retorn de la funció "write", i quan es verificava el volum amb la comanda "hexdump –x" les dades no s'havien modificat.

Estimació temporal

FASE 1

Aproximadament s'ha realitzat aquesta fase en 25 hores, ja que encara no s'havia treballat mai en sistemes de fitxers.

FASE 2

Aproximadament s'ha realitzat aquesta fase en 10 hores.

FASE 3

Aproximadament s'ha realitzat aquesta fase en 5 hores.

FASE 4

Aproximadament s'ha realitzat aquesta fase en 15 hores.

Conclusions

La pràctica de Sistemes operatius avançats ha servit per entendre el funcionament dels sistemes de fitxers FAT16 i EXT2.

Gràcies a realitzar una pràctica i tocar a baix nivell el volum de dades s'ha aconseguit assolir els conceptes i s'ha entès el funcionament intern del sistema.

Per altra banda la pràctica ha servit per veure les limitacions de cadascuna de les tecnologies. Per exemple, FAT16 només pot tenir fitxers de 2^32 bytes, ja que la variable que emmagatzema la mida del fitxer és de 4 bytes. O per exemple la complicació de borrar un fitxer en EXT2, ja que s'ha d'actualitzar l'arbre d'inodes i no deixar cap inode apuntant a un inode buit.

Finalment s'ha aconseguit realitzar la pràctica amb cadascuna de les fases.

Valoracions

La metodologia on demand aplicada ha servit per poder assolir els conceptes de manera més pautada i més lentament.

El fet de realitzar diferents fases, aparentment independents, ha servit per poder estructurar el programa en funció de la necessitat actual. Per aquest motiu es podia centrar els esforços en una funcionalitat en concret, sense pensar en les funcionalitats que s'implementaran en un futur.

Finalment cal recomanar la pràctica a futurs alumnes de l'assignatura, ja que sense ella, no es podrien haver assolit els conceptes de la mateixa manera.