File I/O

Embedded Systems Interfacing
Interfacing

Overview

- DOS Data Structure
  - Sectors and Clusters
  - Master Boot Records/Boot Record
  - File Allocation Table
  - Root Directory
  - Subdirectory File
- File IO Module

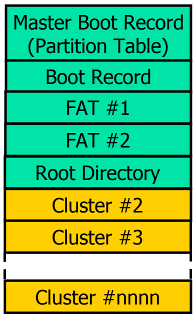
Sectors and Clusters

- Sectors are hardware units with 512 bytes per sector
- Cluster is an file system unit with 2^nx512 bytes per cluster
- Each cluster numbers start at a value of 2 at start of the data space (see future slide)
- Each cluster is an allocation unit in the File Allocation Table or FAT (see future slide)

SD Card Map Structure

- Partition Table (divides media into drives)
- Boot Record (divides drive into data structures)
- 16-Bit File Allocation Table 1 (FAT1)
- 16-Bit File Allocation Table 2 (FAT2)
- Root Directory (fixed number of files)
- Data Area (files and subdirectories)

File System Layout



Master Boot Record (Partition Table)

Boot Record

FAT #1

FAT #2

Root Directory

Cluster #2

Cluster #3

Cluster #nnnn

- Cluster #1 used for Disk Type
- MBR, BR, FATs and Root not covered by FAT Cluster Numbers
- FAT-16 uses two bytes per cluster

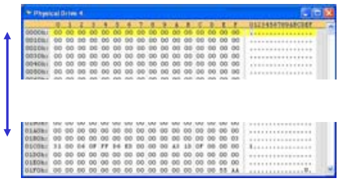
512 MB SD Card Map

Address	Sector	Cluster	Description
0:0000	0	N.A.	Master Boot Record which contains the Partition table
1:DA00	0ED	N.A	Boot Record
1:DC00	0EE	N.A	File Allocation Table 1 with 16-bit FAT entries
2:CE00	167	N.A.	File Allocation Table 2 with 16-bit FAT entries
3:C000	1E0	N.A.	Root Directory with VFAT and directory entries
4:0000	200	2	Bill of Right.txt
4:4000	220	3	MyFile.txt

512 Bytes = 1 Sector[1]

[1] Hardware Term

Partition Table Image



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

01A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
01B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03   ................
01C0h: 31 00 06 0F FF [illegible] ED 00 00 00 A3 1D 0F 00 00 00   1...............
01D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
01E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
01F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
```

Contains
Code for
Bootable
Media

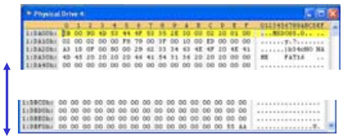
Read/Executed By B.I.O.S.

SD Card Partition Table

Offset	Size	Data	Description
0x1BE	1 Byte	0x00	Not bootable media
0x01BF	1 Byte	0x03	Starting Head (Boot Record)
0x1C0	6 Bits	0B110001	Starting Sector (Boot Record)
0x1C0	10 Bits	0B000000000000	Staring Cylinder (Boot Record)
0x1C2	1 Byte	0x06	System ID: BIGDOS FAT16 partition
0x1C3	1 Byte	0x0F	Ending Head
0x1C4	6 bits	0B111111	Ending Sector
0x1C4	10 bits	0B1101011011	Ending Cylinder
0x1C6	**4 Bytes**	**0x000000ED**	**Relative Sector**
0x1CA	4 Bytes	0X000F1DA3	Total Sectors

Go to LBA 0x000000ED for Boot Record

Boot Record Image



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
1:DA00h: EB 00 90 4D 53 44 4F 53 35 2E 30 00 02 20 01 00   ...MSDOS5.0. ..
1:DA10h: 02 00 02 00 00 F8 79 00 3F 00 10 00 ED 00 00 00   ......y.?.......
1:DA20h: A3 1D 0F 00 80 00 29 62 33 34 63 4E 4F 20 4E 41   ......)b34cNO NA
1:DA30h: 4D 45 20 20 20 20 46 41 54 31 36 20 20 20 00 00   ME    FAT16   ..
1:DA40h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

1:DBC0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DBD0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DBE0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DBF0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
```

Contains
Code for
Bootable
Media

Read/Executed By B.I.O.S.

SD Card Boot Record

Offset	Size	Data	Description
0x0B	2 Bytes	0x200	Bytes per sector (512)
0x0D	1 Byte	0x20	Sectors per cluster (32)
0x0E	2 Bytes	0x0001	Reserved sectors
0x10	1 Byte	0x02	Number of File Allocation Tables
0x11	2 Bytes	0x0200	Root entries (512)
0x13	2 Bytes	0x0000	Small sectors

Standard Information in BIOS Parameter Block

- 512 Bytes/Sector
- 32 Sectors/Cluster
- 1 Reserved Sector
- 2 FATs
- 200 Directory Entries (x 32 Bytes/Entry)

SD Card Boot Record

Offset	Size	Data	Description
0x15	1 Byte	0xF8	Media type
0x16	2 Bytes	0x0079	Sectors per File Allocation Table (121)
0x18	2 Bytes	0x003F	Sectors per track (63)
0x1A	2 Bytes	0x0010	Number of heads (16)
0x1C	4 Bytes	0x000000ED	Hidden sectors (237)
0x20	4 Bytes	0x000F1DA3	Large sectors (990627)
0x24	1 Bytes	0x80	Physical drive number
0x25	1 Byte	0x00	Current head
0x26	1 Byte	0x29	Signature
0x27	4 bytes	0x63343362	Volume serial number
0x2B	11 Bytes	"NO NAME "	Volumn label
0x36	8 Bytes	"FAT16 "	System ID

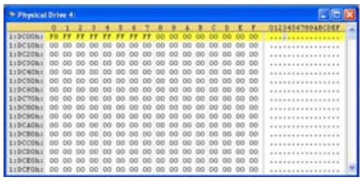
FAT Entry Reference

Entry	Description
0x0000	Free cluster
0x0001	Reserved cluster
0x0002 - 0xFEFF	Used cluster; value points to next cluster
0xFFF0 - 0xFFF6	Reserved values
0xFFF7	Bad cluster
0xFFF8 - 0xFFFF	Last cluster in file

Maximum of 65520 Clusters in FAT-16

Cluster is Software (OS) Term

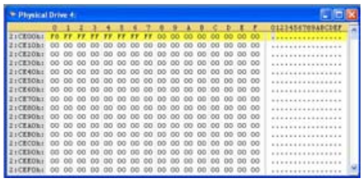
FAT1 Image



Physical Drive 4:

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
1:DC00h:  F0 FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00   ................
1:DC10h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC20h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC30h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC40h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC50h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC60h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC70h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC80h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DC90h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCA0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCB0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCC0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCD0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCE0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1:DCF0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

FAT2 Image

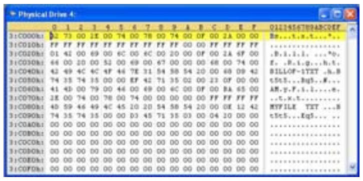


Physical Drive 4:

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
2:CE00h:  F0 FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00   ................
2:CE10h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE20h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE30h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE40h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE50h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE60h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE70h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE80h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CE90h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CEA0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CEB0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CEC0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CED0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CEE0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
2:CEF0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Root Directory Image

- VFAT entries
- Directory entries



Physical Drive 4:

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
3:C000h:  42 73 00 2E 00 74 00 78 00 74 00 0F 00 2A 00 00   Bs...t.x.t...*..
3:C010h:  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF   ................
3:C020h:  01 42 00 69 00 6C 00 6C 00 20 00 0F 00 2A 6F 00   .B.i.l.l. ...*o.
3:C030h:  66 00 20 00 52 00 69 00 67 00 00 00 68 00 74 00   f. .R.i.g...h.t.
3:C040h:  42 49 4C 4C 4F 46 7E 31 54 58 54 20 00 68 09 42   BILLOF~1TXT .h.B
3:C050h:  74 35 74 35 00 00 EF 42 71 35 02 00 23 0F 00 00   t5t5...Bq5..#...
3:C060h:  41 4D 00 79 00 46 00 69 00 6C 00 0F 00 BA 65 00   AM.y.F.i.l....e.
3:C070h:  2E 00 74 00 78 00 74 00 00 00 00 00 FF FF FF FF   ..t.x.t.........
3:C080h:  4D 59 46 49 4C 45 20 20 54 58 54 20 00 0E 12 42   MYFILE  TXT ...B
3:C090h:  74 35 74 35 00 00 D3 45 71 35 03 00 04 20 00 00   t5t5...Eq5... ..
3:C0A0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
3:C0B0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
3:C0C0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
3:C0D0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
3:C0E0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
3:C0F0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

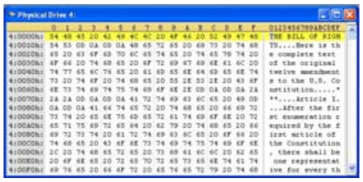
Bill of Rights Directory

Offset	Size	Data	Description
0x00	8 Bytes	"BILLOF~1	8.3 Filename.
0x08	3 Bytes	"TXT	8.3 File extension
0x0B	1 Byte	0x20	Attribute
0x0C	1 Byte	0x00	Reserved
0x0D	1 Byte	0x68	Creation time milliseconds
0x0E	2 Bytes	0x4209	Creation time in hours, minutes and seconds
0x10	2 Bytes	0x3574	Creation date in years since 1980, months, and day.
0x12	2 Bytes	0x3574	Last accessed date
0x14	2 Bytes	0x0000	EA-index
0x16	2 Bytes	0x42EF	Last modified time
0x18	2 Bytes	0x3571	Last modified date
0x1A	2 Bytes	0x0002	First cluster
0x1C	4 Bytes	0x00000F23	File size in bytes.

Long File Reference (VFAT)

Offset	Size	Data	Description
0x00	1 Byte	01	Sequence number
0x01	10 Bytes	"Bill "	Name characters as five UTF-16 characters
0x0B	1 Byte	0x0F	Attribute
0x0C	1 Byte	0x00	Reserved
0x0D	1 Byte	0x2A	Checksum of DOS file name
0x0E	12 Bytes	"of Right"	Name characters as six UTF-16 characters
0x1A	2 Bytes		First cluster which is always 0x0000

First (and only) Sector of Bill of Rights



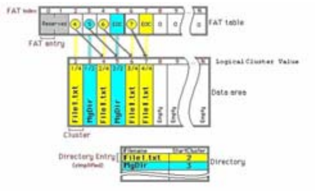
Physical Drive 4:

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
4:0000h:  54 48 45 20 42 49 4C 4C 20 4F 46 20 52 49 47 48   THE BILL OF RIGH
4:0010h:  54 53 0D 0A 0D 0A 48 65 72 65 20 69 73 20 74 68   TS....Here is th
4:0020h:  65 20 63 6F 6D 70 6C 65 74 65 20 74 65 78 74 20   e complete text 
4:0030h:  6F 66 20 74 68 65 20 6F 72 69 67 69 6E 61 6C 20   of the original 
4:0040h:  74 77 65 6C 76 65 20 61 6D 65 6E 64 6D 65 6E 74   twelve amendment
4:0050h:  73 20 74 6F 20 74 68 65 20 55 2E 53 2E 20 43 6F   s to the U.S. Co
4:0060h:  6E 73 74 69 74 75 74 69 6F 6E 2E 0D 0A 0D 0A 2A   nstitution.....*
4:0070h:  2A 2A 0D 0A 0D 0A 41 72 74 69 63 6C 65 20 49 0D   **....Article I.
4:0080h:  0A 0D 0A 41 66 74 65 72 20 74 68 65 20 66 69 72   ...After the fir
4:0090h:  73 74 20 65 6E 75 6D 65 72 61 74 69 6F 6E 20 72   st enumeration r
4:00A0h:  65 71 75 69 72 65 64 20 62 79 20 74 68 65 20 66   equired by the f
4:00B0h:  69 72 73 74 20 61 72 74 69 63 6C 65 20 6F 66 20   irst article of 
4:00C0h:  74 68 65 20 43 6F 6E 73 74 69 74 75 74 69 6F 6E   the Constitution
4:00D0h:  2C 20 74 68 65 72 65 20 73 68 61 6C 6C 20 62 65   , there shall be
4:00E0h:  20 6F 6E 65 20 72 65 70 72 65 73 65 6E 74 61 74    one representat
4:00F0h:  69 76 65 20 66 6F 72 20 65 76 65 72 79 20 74 68   ive for every th
```

File Allocation Table

- FAT12 used for floppy disk
  - Dreamed up by Bill Gates over a weekend
  - Very cryptic and atypical of other OS allocation methods
  - FF 00 10 → 0FF 100
- FAT16 used for small hard drives and SD Cards
- FAT32 used for large hard drives

FAT and Directory Interaction



Get Out Calculator

Find Partition Table 0?

Sector 0, Offset 1BE

Find Boot Record?

0 + $(1C6)_{DWORD}$ = 0x000000ED

Find FAT 1?

0x000000ED + Reserved_{BR} = 0x000000EE

Find FAT 2?

0xEE + Sec/FAT_{BR} = 0x00000176

Get Out Calculator

Find Root Directory?

0x176 + Sector/FAT_{BR} = 0x00000001E0

Data Area?

0x1E0 + Root Entries_{BR}*$\text{(Bytes/Entry)}_{std}$/$\text{(Bytes/Sec)}_{BR}$

= 0x0000000200

Cluster n

0x200 + $(n\text{-}2)_{FAT}$ *$\text{Sector/Cluster}_{BR}$

0x200 + $(3\text{-}2)_{FAT}$ *0x20 = 0x220

Get Out Calculator

First Cluster of File?

0x200+$(\text{File Entry}_{\text{Directory Entry}}\text{-}2)$* $\text{Sector/Cluster}_{BR}$

Next Cluster of File?

0x200 + $(n)_{FAT}$* $\text{Sector/Cluster}_{BR}$

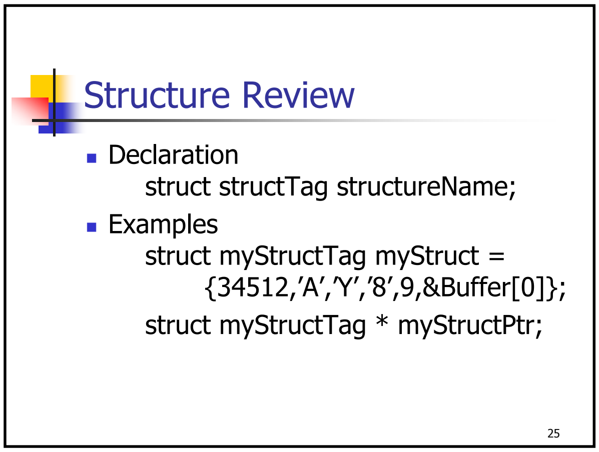
Structure Review

- Template of memory
- struct structTag {

```
    dataType variable name;
    •••
    };
```

- Example

```
    struct myStructTag{
                int myNum;
                char myLetter[4];
                int * myPointer;
                }
```



Structure Review

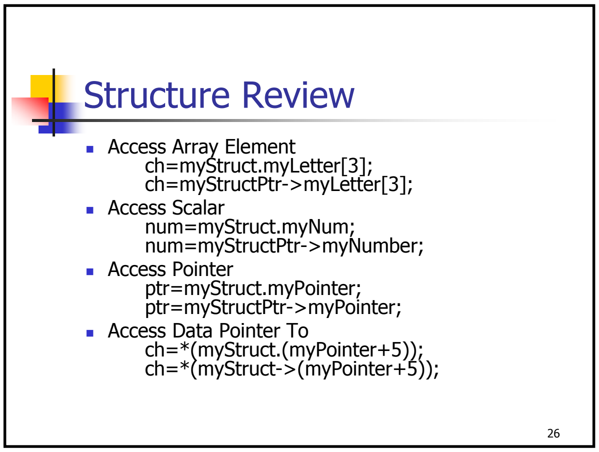
- Declaration

  struct structTag structureName;
- Examples

  struct myStructTag myStruct =

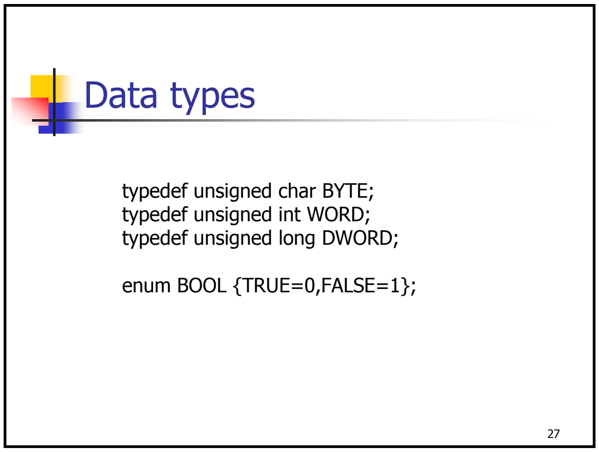
  {34512,’A’,’Y’,’8’,9,&Buffer[0]};

  struct myStructTag * myStructPtr;



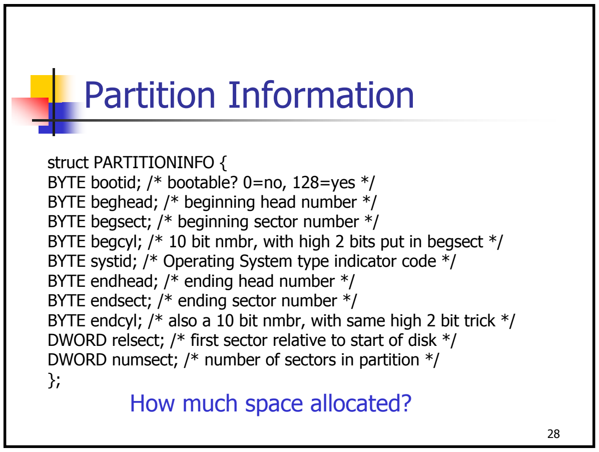
Structure Review

- Access Array Element
  ```
  ch=myStruct.myLetter[3];
  ch=myStructPtr->myLetter[3];
  ```
- Access Scalar
  ```
  num=myStruct.myNum;
  num=myStructPtr->myNumber;
  ```
- Access Pointer
  ```
  ptr=myStruct.myPointer;
  ptr=myStructPtr->myPointer;
  ```
- Access Data Pointer To
  ```
  ch=*(myStruct.(myPointer+5));
  ch=*(myStruct->(myPointer+5));
  ```



Data types

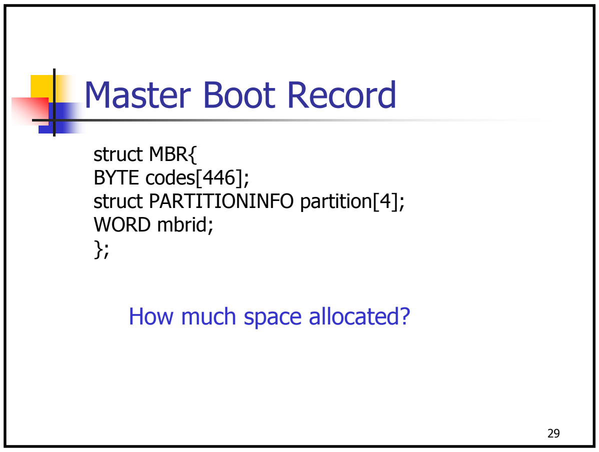
```
typedef unsigned char BYTE;
typedef unsigned int WORD;
typedef unsigned long DWORD;

enum BOOL {TRUE=0,FALSE=1};
```



Partition Information

```
struct PARTITIONINFO {
BYTE bootid; /* bootable? 0=no, 128=yes */
BYTE beghead; /* beginning head number */
BYTE begsect; /* beginning sector number */
BYTE begcyl; /* 10 bit nmbr, with high 2 bits put in begsect */
BYTE systid; /* Operating System type indicator code */
BYTE endhead; /* ending head number */
BYTE endsect; /* ending sector number */
BYTE endcyl; /* also a 10 bit nmbr, with same high 2 bit trick */
DWORD relsect; /* first sector relative to start of disk */
DWORD numsect; /* number of sectors in partition */
};
```

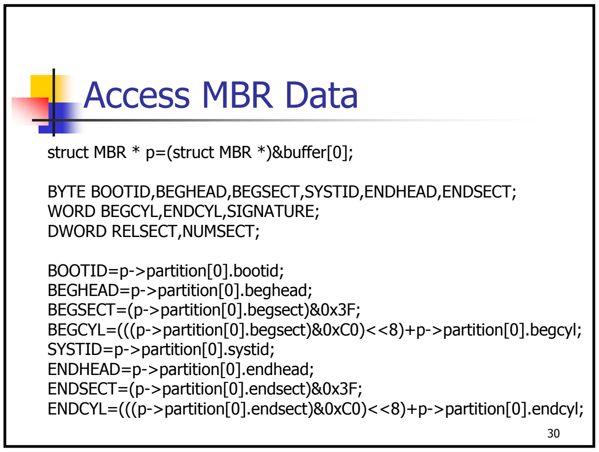
How much space allocated?



Master Boot Record

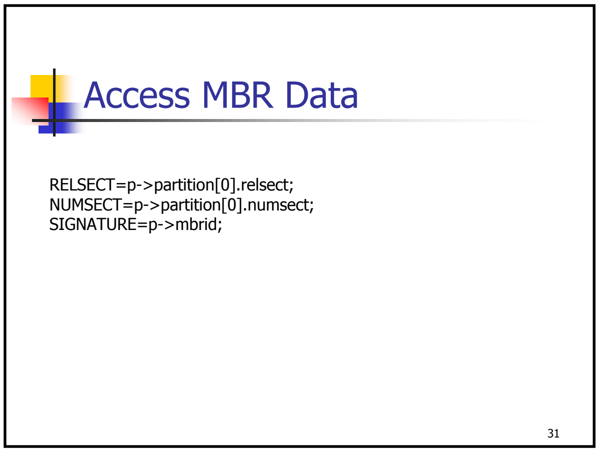
```
struct MBR{
BYTE codes[446];
struct PARTITIONINFO partition[4];
WORD mbrid;
};
```

How much space allocated?



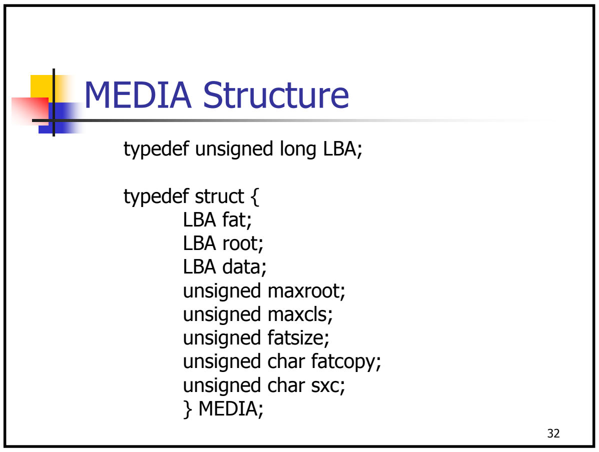
Access MBR Data

```
struct MBR * p=(struct MBR *)&buffer[0];

BYTE BOOTID,BEGHEAD,BEGSECT,SYSTID,ENDHEAD,ENDSECT;
WORD BEGCYL,ENDCYL,SIGNATURE;
DWORD RELSECT,NUMSECT;

BOOTID=p->partition[0].bootid;
BEGHEAD=p->partition[0].beghead;
BEGSECT=(p->partition[0].begsect)&0x3F;
BEGCYL=(((p->partition[0].begsect)&0xC0)<<8)+p->partition[0].begcyl;
SYSTID=p->partition[0].systid;
ENDHEAD=p->partition[0].endhead;
ENDSECT=(p->partition[0].endsect)&0x3F;
ENDCYL=(((p->partition[0].endsect)&0xC0)<<8)+p->partition[0].endcyl;
```



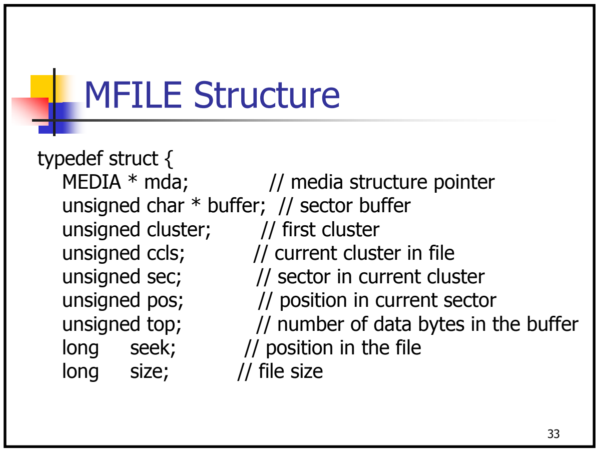
Access MBR Data

```
RELSECT=p->partition[0].relsect;
NUMSECT=p->partition[0].numsect;
SIGNATURE=p->mbrid;
```



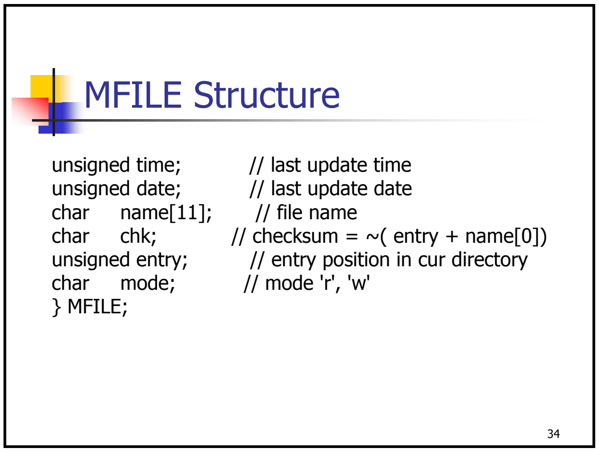
MEDIA Structure

```
typedef unsigned long LBA;

typedef struct {
        LBA fat;
        LBA root;
        LBA data;
        unsigned maxroot;
        unsigned maxcls;
        unsigned fatsize;
        unsigned char fatcopy;
        unsigned char sxc;
        } MEDIA;
```



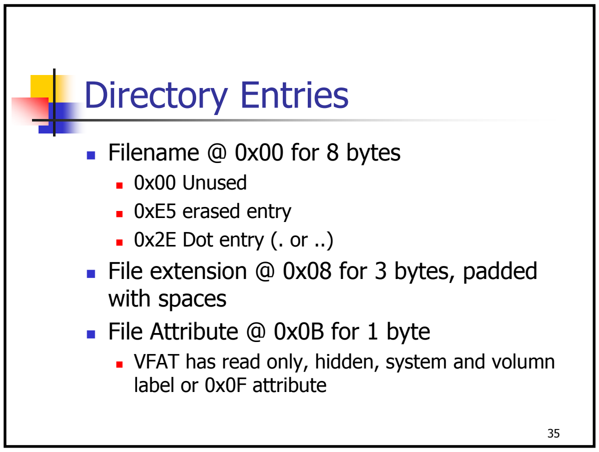
MFILE Structure

```
typedef struct {
    MEDIA * mda;              // media structure pointer
    unsigned char * buffer;  // sector buffer
    unsigned cluster;        // first cluster
    unsigned ccls;           // current cluster in file
    unsigned sec;            // sector in current cluster
    unsigned pos;            // position in current sector
    unsigned top;            // number of data bytes in the buffer
    long     seek;           // position in the file
    long     size;           // file size
```



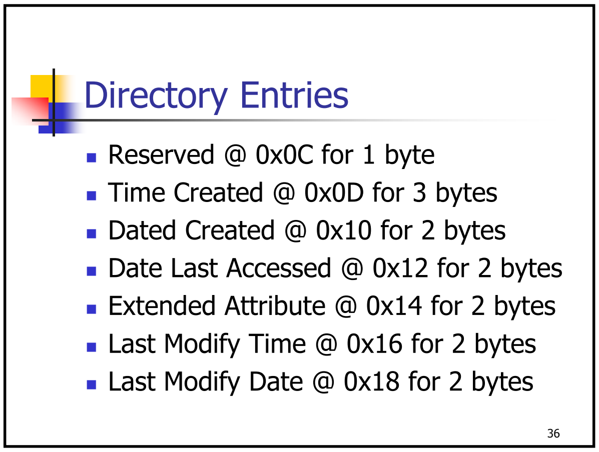
MFILE Structure

```
unsigned time;          // last update time
unsigned date;          // last update date
char     name[11];      // file name
char     chk;         // checksum = ~( entry + name[0])
unsigned entry;         // entry position in cur directory
char     mode;          // mode 'r', 'w'
} MFILE;
```



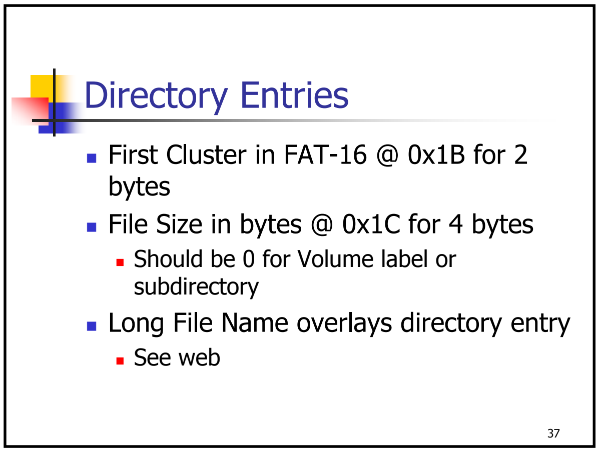
Directory Entries

- Filename @ 0x00 for 8 bytes
  - 0x00 Unused
  - 0xE5 erased entry
  - 0x2E Dot entry (. or ..)
- File extension @ 0x08 for 3 bytes, padded with spaces
- File Attribute @ 0x0B for 1 byte
  - VFAT has read only, hidden, system and volumn label or 0x0F attribute



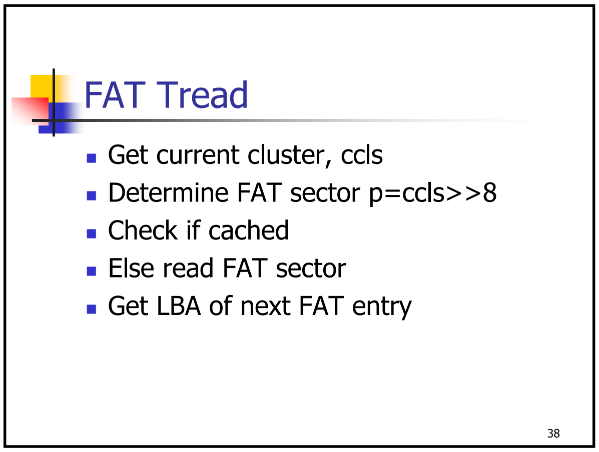
Directory Entries

- Reserved @ 0x0C for 1 byte
- Time Created @ 0x0D for 3 bytes
- Dated Created @ 0x10 for 2 bytes
- Date Last Accessed @ 0x12 for 2 bytes
- Extended Attribute @ 0x14 for 2 bytes
- Last Modify Time @ 0x16 for 2 bytes
- Last Modify Date @ 0x18 for 2 bytes



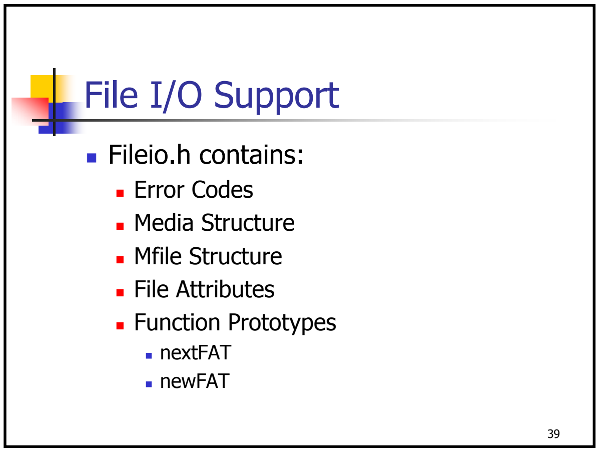
Directory Entries

- First Cluster in FAT-16 @ 0x1B for 2 bytes
- File Size in bytes @ 0x1C for 4 bytes
  - Should be 0 for Volume label or subdirectory
- Long File Name overlays directory entry
  - See web



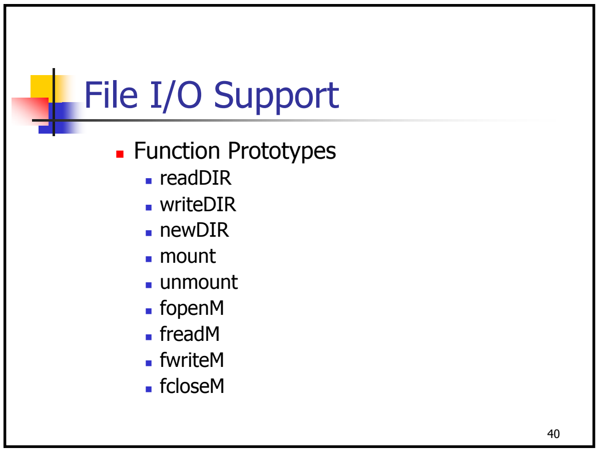
FAT Tread

- Get current cluster, ccls
- Determine FAT sector p=ccls>>8
- Check if cached
- Else read FAT sector
- Get LBA of next FAT entry



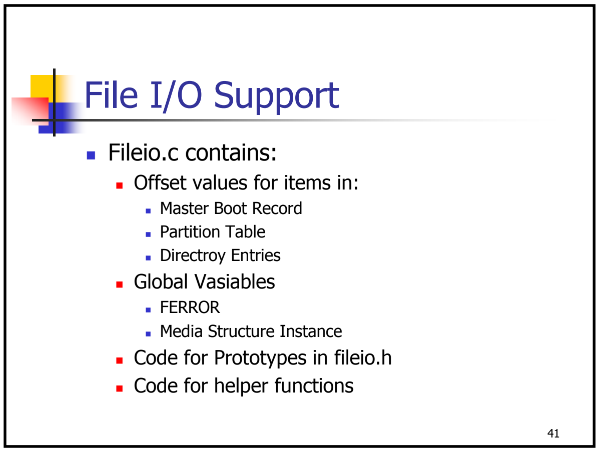
File I/O Support

- Fileio.h contains:
  - Error Codes
  - Media Structure
  - Mfile Structure
  - File Attributes
  - Function Prototypes
    - nextFAT
    - newFAT



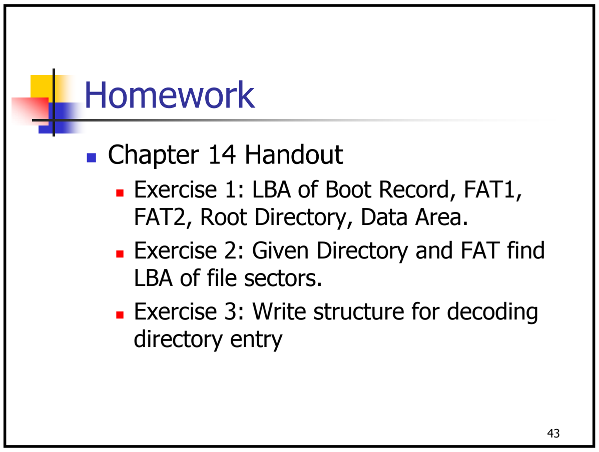
File I/O Support

- Function Prototypes
  - readDIR
  - writeDIR
  - newDIR
  - mount
  - unmount
  - fopenM
  - freadM
  - fwriteM
  - fcloseM



File I/O Support

- Fileio.c contains:
  - Offset values for items in:
    - Master Boot Record
    - Partition Table
    - Directroy Entries
  - Global Vasiables
    - FERROR
    - Media Structure Instance
  - Code for Prototypes in fileio.h
  - Code for helper functions



Homework

- Chapter 14 Handout
  - Exercise 1: LBA of Boot Record, FAT1, FAT2, Root Directory, Data Area.
  - Exercise 2: Given Directory and FAT find LBA of file sectors.
  - Exercise 3: Write structure for decoding directory entry