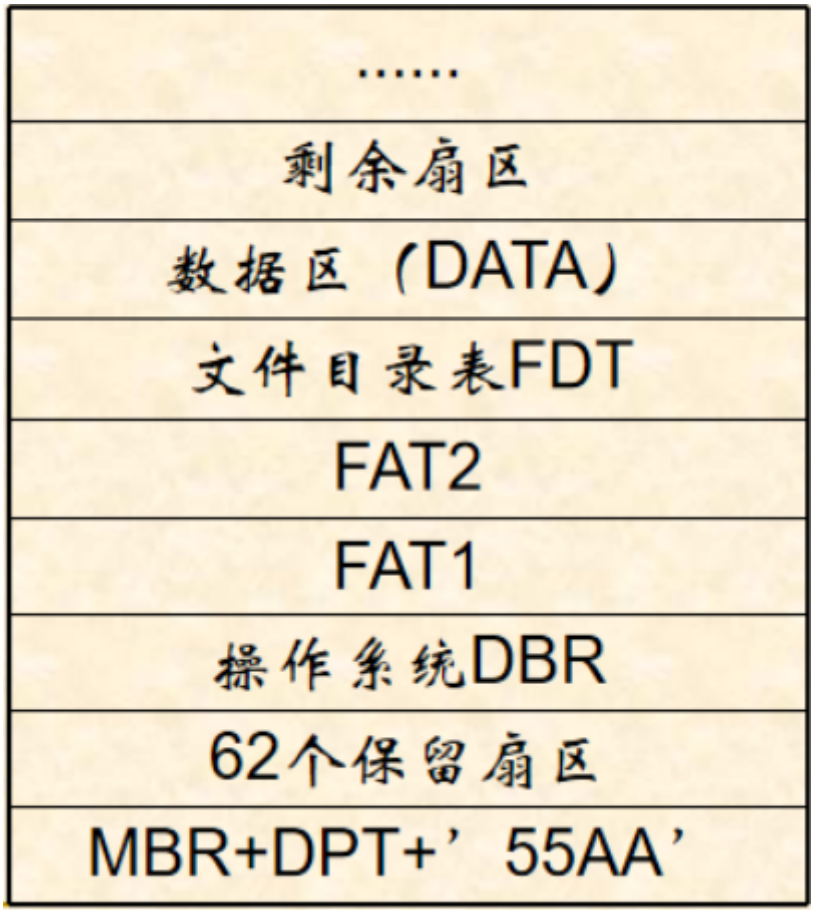
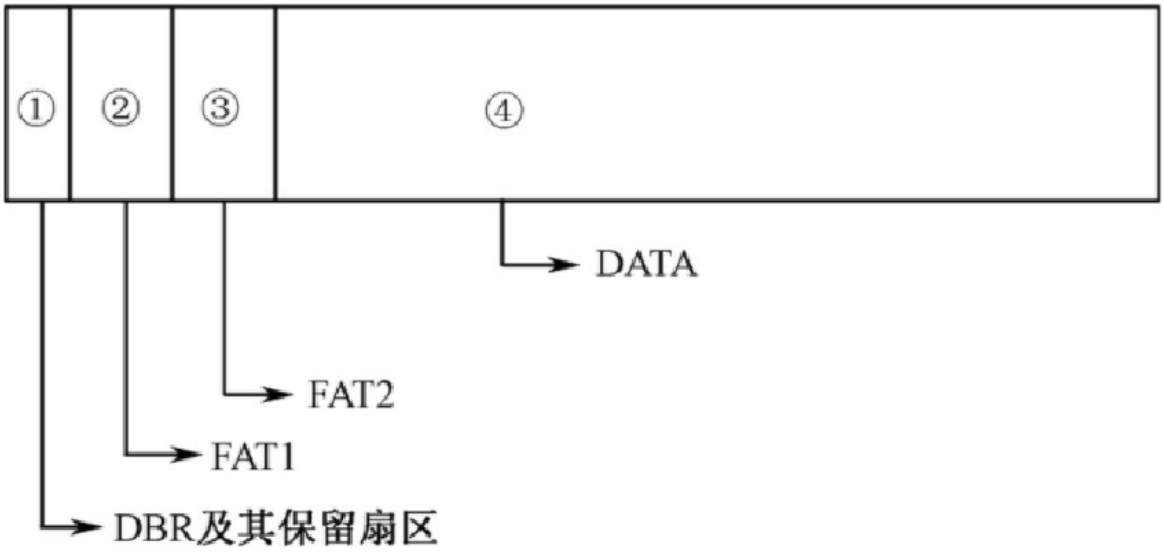
# Winhex

## FAT32文件系统结构

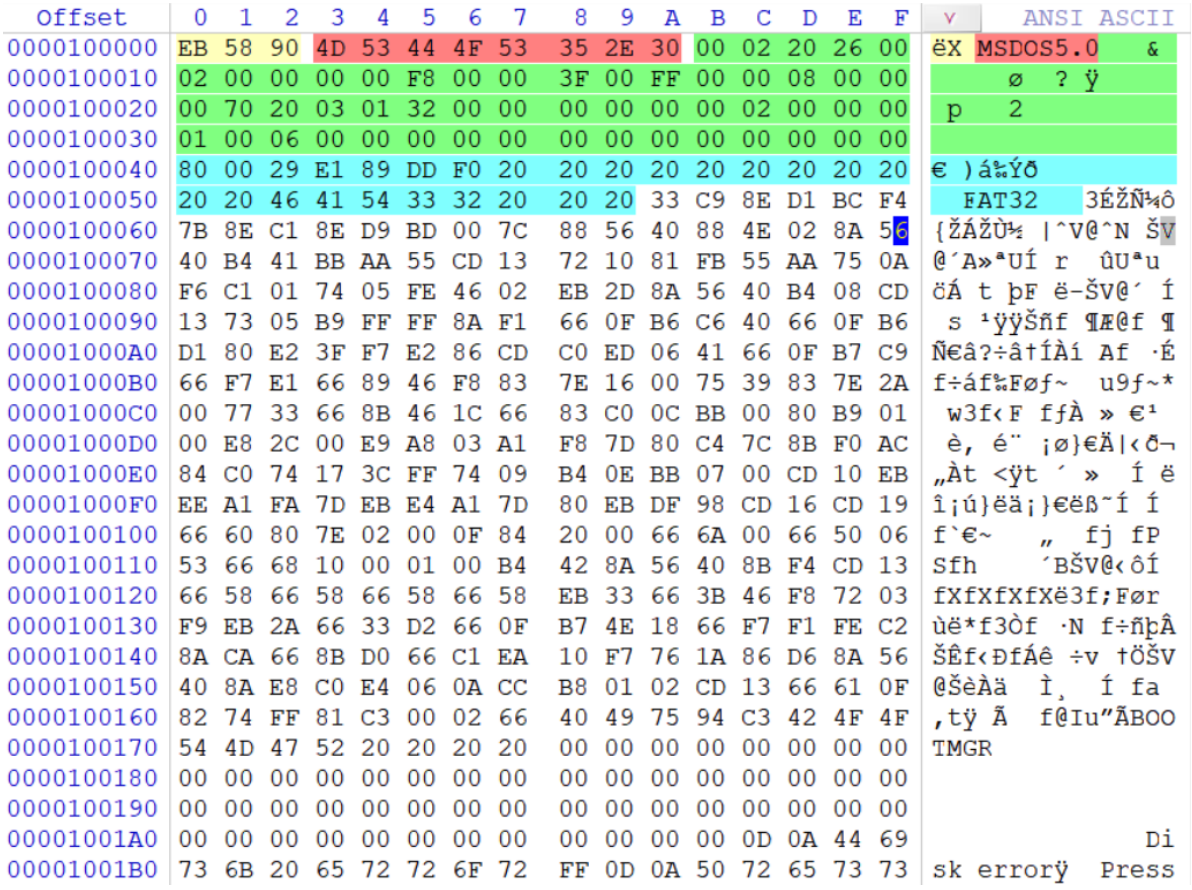



## DBR结构

| FAT32分区上DBR中各部分的位置划分 | | |
|---|---|---|
| 字节位移 | 字段长度 | 字段名 |
| 0x00 | 3个字节 | 跳转指令 |
| 0x03 | 8个字节 | 厂商标志和os版本号 |
| 0x0B | 53个字节 | BPB |
| 0x40 | 26个字节 | 扩展BPB |
| 0x5A | 420个字节 | 引导程序代码 |
| 0x01FE | 2个字节 | 有效结束标志 |

| 字节偏移 | 字段长度（字节） | 字段内容及含义 | 字节偏移 | 字段长度（字节） | 字段内容及含义 |
|---|---|---|---|---|---|
| 0x0B | 2 | 每扇区字节数 | 0x28 | 2 | 标记 |
| 0x0D | 1 | 每簇扇区数 | 0x2A | 2 | 版本 |
| 0x0E | 2 | DOS 保留扇区数 | 0x2C | 4 | 根目录首簇号 |
| 0x10 | 1 | FAT 表个数 | 0x30 | 2 | 文件系统信息扇区号 |
| 0x11 | 2 | 未用 | 0x32 | 2 | DBR 备份扇区号 |
| 0x13 | 2 | 未用 | 0x34 | 12 | 保留 |
| 0x15 | 1 | 介质描述符（十六进制） | 0x40 | 1 | BIOS 驱动器号 |
| 0x16 | 2 | 未用 | 0x41 | 1 | 未用 |
| 0x18 | 2 | 每磁道扇区数 | 0x42 | 1 | 扩展引导标记 |
| 0x1A | 2 | 磁头数 | 0x43 | 4 | 卷序列号 |
| 0x1C | 4 | 隐藏扇区 | 0x47 | 11 | 卷标 |
| 0x20 | 4 | 该分区的扇区总数 | 0x52 | 8 | 文件系统类型 |
| 0x24 | 4 | 每 FAT 扇区数 | | | |



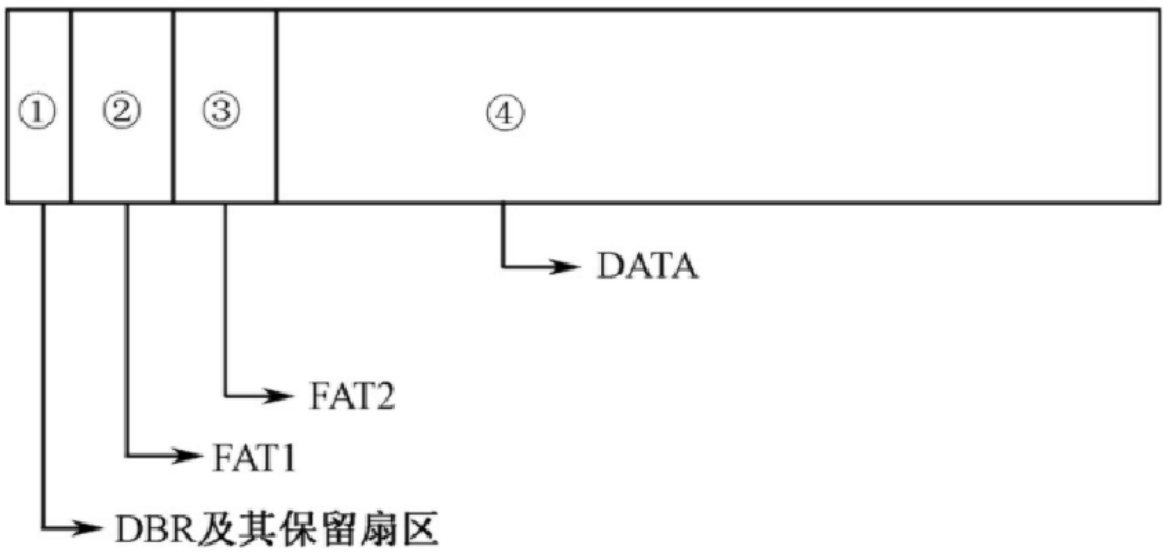
| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | ANSI ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000100000 | EB | 58 | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 20 | 26 | 00 | ëX MSDOS5.0 & |
| 0000100010 | 02 | 00 | 00 | 00 | 00 | F8 | 00 | 00 | 3F | 00 | FF | 00 | 00 | 08 | 00 | 00 | ø ? ÿ |
| 0000100020 | 00 | 70 | 20 | 03 | 01 | 32 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 00 | 00 | p 2 |
| 0000100030 | 01 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0000100040 | 80 | 00 | 29 | E1 | 89 | DD | F0 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | € )á‰Ýð |
| 0000100050 | 20 | 20 | 46 | 41 | 54 | 33 | 32 | 20 | 20 | 20 | 33 | C9 | 8E | D1 | BC | F4 | FAT32 3ÉŽÑ¼ô |
| 0000100060 | 7B | 8E | C1 | 8E | D9 | BD | 00 | 7C | 88 | 56 | 40 | 88 | 4E | 02 | 8A | 56 | {ŽÁŽÙ½ \|ˆV@ˆN ŠV |
| 0000100070 | 40 | B4 | 41 | BB | AA | 55 | CD | 13 | 72 | 10 | 81 | FB | 55 | AA | 75 | 0A | @´A»ªUÍ r ûUªu |
| 0000100080 | F6 | C1 | 01 | 74 | 05 | FE | 46 | 02 | EB | 2D | 8A | 56 | 40 | B4 | 08 | CD | öÁ t þF ë-ŠV@´ Í |
| 0000100090 | 13 | 73 | 05 | B9 | FF | FF | 8A | F1 | 66 | 0F | B6 | C6 | 40 | 66 | 0F | B6 | s ¹ÿÿŠñf ¶Æ@f ¶ |
| 00001000A0 | D1 | 80 | E2 | 3F | F7 | E2 | 86 | CD | C0 | ED | 06 | 41 | 66 | 0F | B7 | C9 | Ñ€â?÷â†ÍÀí Af ·É |
| 00001000B0 | 66 | F7 | E1 | 66 | 89 | 46 | F8 | 83 | 7E | 16 | 00 | 75 | 39 | 83 | 7E | 2A | f÷áf‰Føƒ~ u9ƒ~* |
| 00001000C0 | 00 | 77 | 33 | 66 | 8B | 46 | 1C | 66 | 83 | C0 | 0C | BB | 00 | 80 | B9 | 01 | w3f‹F fƒÀ » €¹ |
| 00001000D0 | 00 | E8 | 2C | 00 | E9 | A8 | 03 | A1 | F8 | 7D | 80 | C4 | 7C | 8B | F0 | AC | è, é¨ ¡ø}€Ä\|‹ð¬ |
| 00001000E0 | 84 | C0 | 74 | 17 | 3C | FF | 74 | 09 | B4 | 0E | BB | 07 | 00 | CD | 10 | EB | „Àt <ÿt ´ » Í ë |
| 00001000F0 | EE | A1 | FA | 7D | EB | E4 | A1 | 7D | 80 | EB | DF | 98 | CD | 16 | CD | 19 | î¡ú}ëä¡}€ëß˜Í Í |
| 0000100100 | 66 | 60 | 80 | 7E | 02 | 00 | 0F | 84 | 20 | 00 | 66 | 6A | 00 | 66 | 50 | 06 | f`€~ „ fj fP |
| 0000100110 | 53 | 66 | 68 | 10 | 00 | 01 | 00 | B4 | 42 | 8A | 56 | 40 | 8B | F4 | CD | 13 | Sfh ´BŠV@‹ôÍ |
| 0000100120 | 66 | 58 | 66 | 58 | 66 | 58 | 66 | 58 | EB | 33 | 66 | 3B | 46 | F8 | 72 | 03 | fXfXfXfXë3f;Før |
| 0000100130 | F9 | EB | 2A | 66 | 33 | D2 | 66 | 0F | B7 | 4E | 18 | 66 | F7 | F1 | FE | C2 | ùë*f3Òf ·N f÷ñþÂ |
| 0000100140 | 8A | CA | 66 | 8B | D0 | 66 | C1 | EA | 10 | F7 | 76 | 1A | 86 | D6 | 8A | 56 | ŠÊf‹ÐfÁê ÷v †ÖŠV |
| 0000100150 | 40 | 8A | E8 | C0 | E4 | 06 | 0A | CC | B8 | 01 | 02 | CD | 13 | 66 | 61 | 0F | @ŠèÀä Ì¸ Í fa |
| 0000100160 | 82 | 74 | FF | 81 | C3 | 00 | 02 | 66 | 40 | 49 | 75 | 94 | C3 | 42 | 4F | 4F | ‚tÿ Ã f@Iu”ÃBOO |
| 0000100170 | 54 | 4D | 47 | 52 | 20 | 20 | 20 | 20 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | TMGR |
| 0000100180 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0000100190 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00001001A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 0D | 0A | 44 | 69 | Di |
| 00001001B0 | 73 | 6B | 20 | 65 | 72 | 72 | 6F | 72 | FF | 0D | 0A | 50 | 72 | 65 | 73 | 73 | sk errorÿ Press |

| **DBR信息** | |
|---|---|
| 每扇区字节数 | 512 |
| 每簇扇区数 | 32 |
| 保留扇区数 | 38 |
| 每FAT扇区数 | 12801 |
| 根目录起始簇 | 2 |

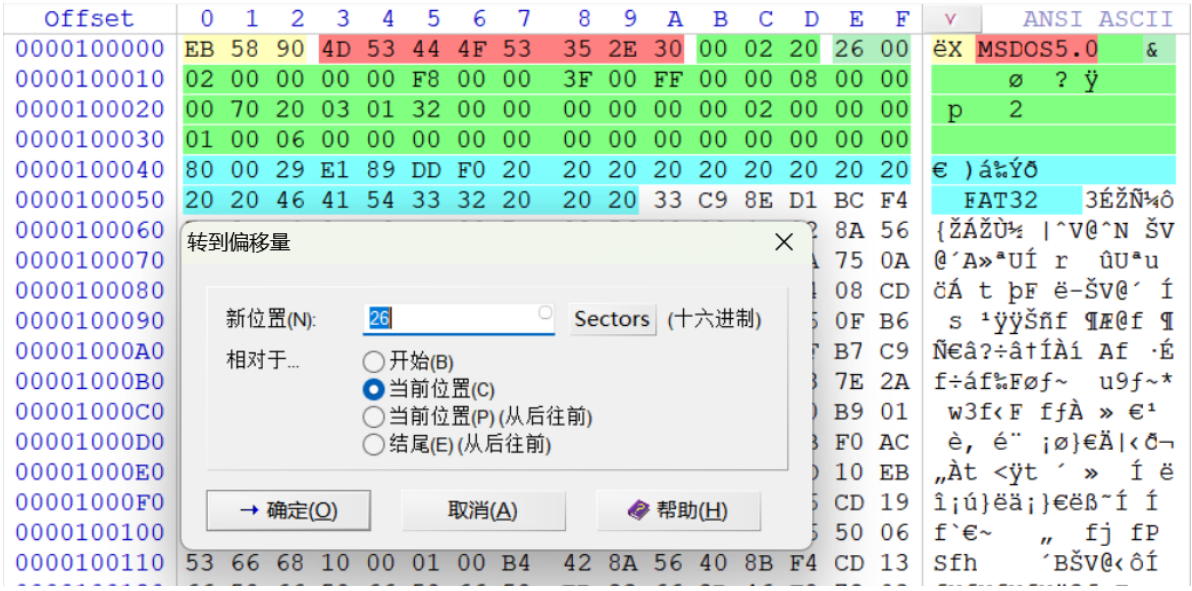
Boot Sector FAT32, 基本偏移: 100000

| Offset | 标题 | 数值 |
|---|---|---|
| 100000 | JMP instruction | EB 58 90 |
| 100003 | OEM | MSDOS5.0 |
| | BIOS Parameter Block | |
| 10000B | Bytes per sector | 512 |
| 10000D | Sectors per cluster | 32 |
| 10000E | Reserved sectors | 38 |
| 100010 | Number of FATs | 2 |
| 100011 | Root entries (unused) | 0 |
| 100013 | Sectors (on small volumes) | 0 |
| 100015 | Media descriptor (hex) | F8 |
| 100016 | Sectors per FAT (small vol.) | 0 |
| 100018 | Sectors per track | 63 |
| 10001A | Heads | 255 |
| 10001C | Hidden sectors | 2,048 |
| 100020 | Sectors (on large volumes) | 52,457,472 |
| | FAT32 Section | |
| 100024 | Sectors per FAT | 12,801 |
| 100028 | Extended flags | 0 |
| 100028 | FAT mirroring disabled? | 0 |
| 10002A | Version (usually 0) | 0 |
| 10002C | Root dir 1st cluster | 2 |
| 100030 | FSInfo sector | 1 |
| 100032 | Backup boot sector | 6 |
| 100034 | (Reserved) | 00 00 00 00 00 00 00 00 00 00 00 00 |
| 100040 | BIOS drive (hex, HD=8x) | 80 |
| 100041 | (Unused) | 0 |
| 100042 | Ext. boot signature (29h) | 29 |
| 100043 | Volume serial number (decimal) | 2,285,822,631 |
| 100043 | Volume serial number (hex) | A7 E2 3E 88 |
| 100047 | Volume label | |
| 100052 | File system | FAT32 |

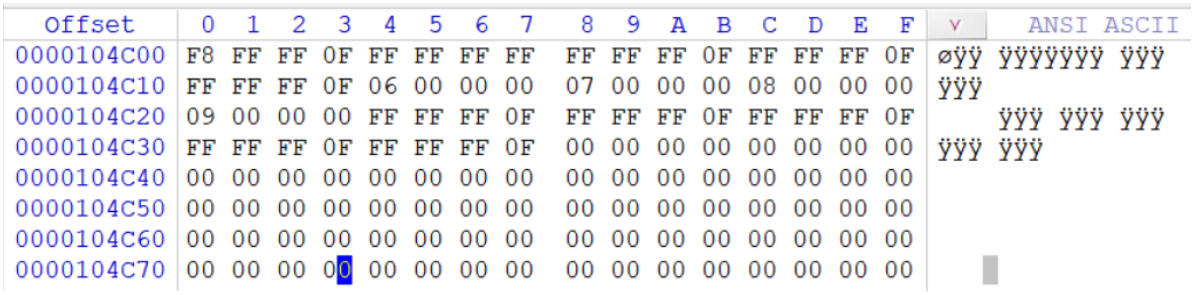
# 定位FAT1




根据FAT32文件系统的结构，我们可知，在FAT文件系统中FAT表是在保留扇区之后，FAT1表前面的都是保留扇区，所以很容易得到FAT1表的位置需要从DBR跳转38个sector。

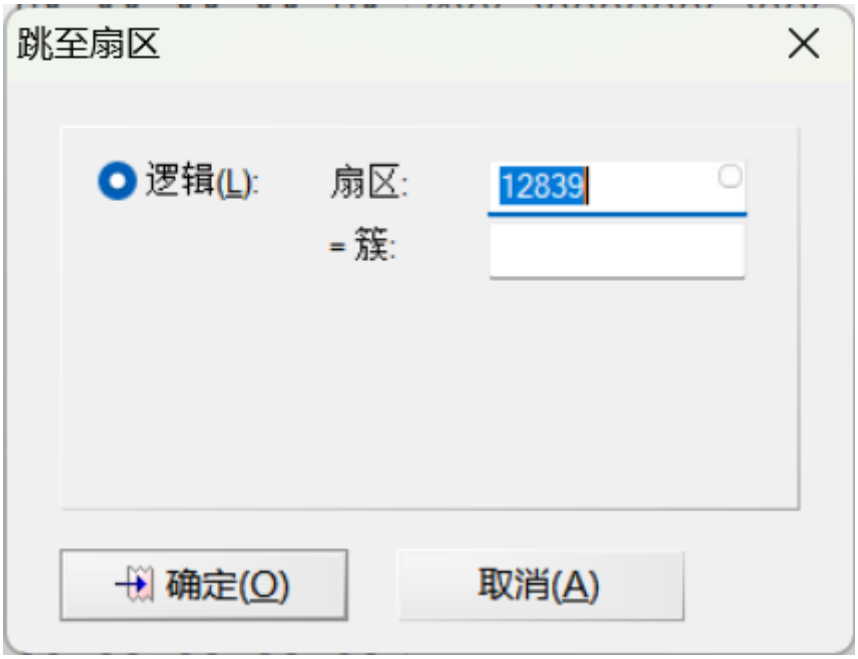



FAT1由DBT偏移0x26，即38个扇区




# 定位FAT2

由FAT1偏移从DBR中解析得到的每FAT扇区数：38+12801=12839




注意区分输入的是十进制还是十六进制，这里是十进制

```
000644E00  F8 FF FF 0F FF FF FF FF  FF FF FF 0F FF FF FF 0F  øÿÿ ÿÿÿÿÿÿÿÿ ÿÿÿ
000644E10  FF FF FF 0F 06 00 00 00  07 00 00 00 08 00 00 00  ÿÿÿ
000644E20  09 00 00 00 FF FF FF 0F  FF FF FF 0F FF FF FF 0F      ÿÿÿ ÿÿÿ ÿÿÿ
000644E30  FF FF FF 0F FF FF FF 0F  00 00 00 00 00 00 00 00  ÿÿÿ ÿÿÿ
000644E40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000644E50  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000644E60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000644E70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

找到FAT2

# 定位根目录

FAT2表过后就是rootdir的内容，也就是需要跳转的sector=12801*2+38=25640个sector，从当前分区。

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F   ANSI ASCII
000C85000 CF B5 CD B3 20 20 20 20  20 20 20 08 00 00 00 00  Ïµí³
000C85010 00 00 00 00 00 00 9A 8A  8F 56 00 00 00 00 00 00        šŠ V
000C85020 46 4F 4C 44 45 52 20 20  20 20 20 10 08 4B A7 8A  FOLDER       K§Š
000C85030 8F 56 8F 56 00 00 26 71  8F 56 03 00 00 00 00 00   V V  &q V
000C85040 42 74 00 78 00 74 00 00  00 FF FF 0F 00 D4 FF FF  Bt x t   ÿÿ  Ôÿÿ
000C85050 FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿ  ÿÿÿÿ
000C85060 01 6C 00 6F 00 6E 00 67  00 46 00 0F 00 D4 69 00   l o n g F   Ôi
000C85070 6C 00 65 00 4E 00 61 00  6D 00 00 00 65 00 2E 00  l e N a m   e .
000C85080 4C 4F 4E 47 46 49 7E 31  54 58 54 20 00 4E A7 8A  LONGFI~1TXT  N§Š
000C85090 8F 56 8F 56 00 00 D5 79  8F 56 04 00 40 17 00 00   V V  Õy V  @
000C850A0 54 45 53 54 20 20 20 20  54 58 54 20 18 51 A7 8A  TEST    TXT  Q§Š
000C850B0 8F 56 8F 56 00 00 77 71  8F 56 05 00 50 2A 01 00   V V  wq V  P*
000C850C0 41 4B 6D D5 8B 87 65 2C  67 2E 00 0F 00 8C 74 00  AKmÕ‹‡e,g.   Œt
000C850D0 78 00 74 00 00 00 FF FF  FF FF 00 00 FF FF FF FF  x t   ÿÿÿÿ  ÿÿÿÿ
000C850E0 B2 E2 CA D4 CE C4 B1 BE  54 58 54 20 00 54 A7 8A  ²âÊÔÎÄ±¾TXT  T§Š
000C850F0 8F 56 8F 56 00 00 80 71  8F 56 0A 00 30 02 00 00   V V  €q V  0
000C85100 42 20 00 49 00 6E 00 66  00 6F 00 0F 00 72 72 00  B  I n f o   rr
000C85110 6D 00 61 00 74 00 69 00  6F 00 00 00 6E 00 00 00  m a t i o   n
000C85120 01 53 00 79 00 73 00 74  00 65 00 0F 00 72 6D 00   S y s t e   rm
000C85130 20 00 56 00 6F 00 6C 00  75 00 00 00 6D 00 65 00    V o l u   m e
000C85140 53 59 53 54 45 4D 7E 31  20 20 20 16 00 39 0C A2  SYSTEM~1     9 ¢
000C85150 8F 56 8F 56 00 00 0D A2  8F 56 0B 00 00 00 00 00   V V   ¢ V
000C85160 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000C85170 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

# 解析文件信息

## 短文件

| FAT32短文件目录项32个字节的表示定义 | | | | | | |
|---|---|---|---|---|---|---|
| 字节偏移 | 字节数 | | 定义 | | 字节数 | |
| 0x0~0x7 | 8 | | 文件名 | 0xD | 1 | 创建时间的10毫秒位 |
| 0x8~0xA | 3 | | 扩展名 | 0xE~0xF | 2 | 文件创建时间 |
| 0xB* | 1 | 属性字节 | 00000000(读写) | 0x10~0x11 | 2 | 文件创建日期 |
| | | | 00000001(只读) | 0x12~0x13 | 2 | 文件最后访问日期 |
| | | | 00000010(隐藏) | 0x14~0x15 | 2 | 文件起始簇号的高16位 |
| | | | 00000100(系统) | 0x16~0x17 | 2 | 文件的最近修改时间 |
| | | | 00001000(卷标) | 0x18~0x19 | 2 | 文件的最近修改日期 |
| | | | 00010000(子目录) | 0x1A~0x1B | 2 | 文件起始簇号的低16位 |
| | | | 00100000(归档) | 0x1C~0x1F | 4 | 表示文件的长度 |
| 0xC | 1 | | 系统保留 | | | |

test.txt


| test.txt | |
|---|---|
| 文件名 | test |
| 扩展名 | txt |
| 属性 | 0x20（归档） |
| 起始簇号 | 5 |
| 文件长度 | 0x12A50 byte = 76368 byte = 75kB |

## 长文件

FAT32长文件目录项32个字节的表示定义

| 字节偏移 | 字节数 | 定义 | | | 字节偏移 | 字节数 | 定义 |
|---|---|---|---|---|---|---|---|
| 0x0 | 1 | 属性字节位意义 | 7 | 保留未用 | 0xB | 1 | 长文件名目录项标志，取值0FH |
| | | | 6 | “1” 最后一个目录项 | 0xC | 1 | 系统保留 |
| | | | 5 | 保留未用 | 0xD | 1 | 校验值(根据短文件名计算得出) |
| | | | 4 | 顺序号数值 | 0xE~0x19 | 12 | 长文件名unicode码 |
| | | | 3 | | 0x1A~0x1B | 2 | 文件起始簇号(常置0) |
| | | | 2 | | 0x1C~0x1F | 4 | 长文件名unicode码 |
| | | | 1 | | | | |
| | | | 0 | | | | |
| 0x1~0xA | 10 | 长文件名unicode码 | | | | | |

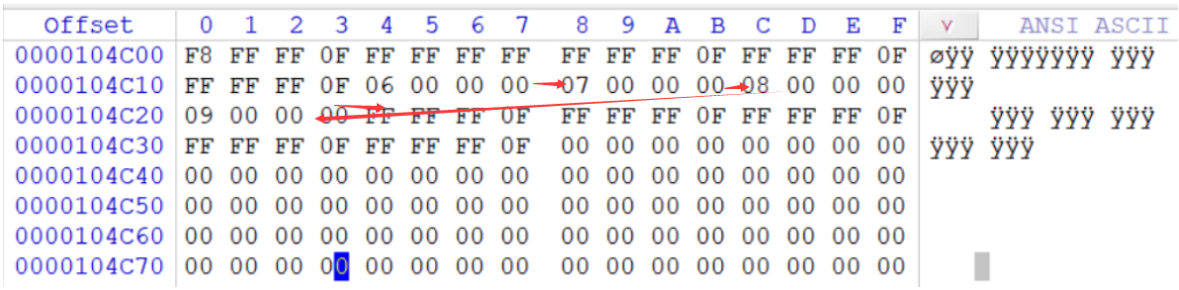
```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F   ANSI ASCII
000C85030 8F 56 8F 56 00 00 26 71  8F 56 03 00 00 00 00 00   V V  &q V
000C85040 42 74 00 78 00 74 00 00  00 FF FF 0F 00 D4 FF FF  Bt x t   ÿÿ  Ôÿÿ
000C85050 FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿ  ÿÿÿÿ
000C85060 01 6C 00 6F 00 6E 00 67  00 46 00 0F 00 D4 69 00   l o n g F   Ôi
000C85070 6C 00 65 00 4E 00 61 00  6D 00 00 00 65 00 2E 00  l e N a m   e .
000C85080 4C 4F 4E 47 46 49 7E 31  54 58 54 20 00 4E A7 8A  LONGFI~1TXT  N§Š
000C85090 8F 56 8F 56 00 00 D5 79  8F 56 04 00 40 17 00 00  V V  Õy V  @
000C850A0 54 45 53 54 20 20 20 20  54 58 54 20 18 51 A7 8A  TEST    TXT  Q§Š
```

```
000C85050 FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿ  ÿÿÿÿ
000C85060 01 6C 00 6F 00 6E 00 67  00 46 00 0F 00 D4 69 00   l o n g F   Ôi
000C85070 6C 00 65 00 4E 00 61 00  6D 00 00 00 65 00 2E 00  l e N a m   e .
000C85080 4C 4F 4E 47 46 49 7E 31  54 58 54 20 00 4E A7 8A  LONGFI~1TXT  N§Š
000C85090 8F 56 8F 56 00 00 D5 79  8F 56 04 00 40 17 00 00  V V  Õy V  @
000C850A0 54 45 53 54 20 20 20 20  54 58 54 20 18 51 A7 8A  TEST    TXT  Q§Š
```

# 簇号链

test.txt


test.txt的起始簇号位5，查找FAT表



| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | ANSI ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000104C00 | F8 | FF | FF | 0F | FF | FF | FF | FF | FF | FF | FF | 0F | FF | FF | FF | 0F | øÿÿ ÿÿÿÿÿÿÿÿ |
| 0000104C10 | FF | FF | FF | 0F | 06 | 00 | 00 | 00 | 07 | 00 | 00 | 00 | 08 | 00 | 00 | 00 | ÿÿÿ |
| 0000104C20 | 09 | 00 | 00 | 00 | FF | FF | FF | 0F | FF | FF | FF | 0F | FF | FF | FF | 0F | ÿÿÿ ÿÿÿ ÿÿÿ |
| 0000104C30 | FF | FF | FF | 0F | FF | FF | FF | 0F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ÿÿÿ ÿÿÿ |
| 0000104C40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0000104C50 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0000104C60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0000104C70 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |

簇号链为：5 -> 6 -> 7 -> 8 -> 9

# 转入data区

由于rootdir默认是从2号簇开始，所以要找到test.txt的数据需要从rootdir起始位置偏移5-2=3个簇，即3*32=0x60个扇区。

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | ANSI ASCII | UTF-8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000C91000 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91010 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91020 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91030 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91040 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91050 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91060 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91070 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91080 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C91090 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C910A0 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C910B0 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C910C0 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |
| 000C910D0 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | 74 | 65 | 73 | 74 | testtesttesttest | testtesttesttest |

# 编程实现

## 声明结构体

```
struct BPB {

uint8_t bytes_per_sector[2];//扇区字节数

uint8_t secotrs_per_cluster;//每簇扇区数

uint8_t reserve_sectors[2];//包括DBR自己在内的FAT之前的扇区个数,即保留扇区数

uint8_t FATnum;//FAT个数，一般为2

uint8_t unimportant1[11];

uint8_t DBR_LBA[4];//该分区的DBR所在的相对扇区号，如果是扩展分区，是相对于扩展分区首

的

uint8_t totalsectors[4];//本分区的总扇区数

uint8_t sectors_per_FAT[4];//每个FAT的扇区数

uint8_t unimportant2[4];

uint8_t root_cluster_number[4];//根目录簇号
```

```
uint8_t file_info[2];

uint8_t backup_DBR[2];//备份引导扇区的相对于DBR的扇区号，一般为6，内容和DBR一模一样

uint8_t zero1[12];

uint8_t extBPB[26];//扩展BPB

};

struct DBR {

uint8_t jumpcode[3];//EB 58 90

uint8_t OEM[8];//OEM代号

BPB bpb;

uint8_t osboot[422];//引导代码和55AA

};

struct shortfile {

uint8_t FileName[8];//文件名

uint8_t ExtendName[3];//扩展名

uint8_t attributeOfFile;//属性字节

uint8_t SystemReserve;//系统保留

uint8_t CreateTime_ms;//创建时间的10毫秒位

uint8_t CreateTime[2];//创建时间

uint8_t CreateDate[2];//创建日期

uint8_t LastAccess[2];//最后访问日期

uint8_t HighCluster[2];//文件起始簇号高16位

uint8_t LastModifyTime[2];//最近修改时间

uint8_t LastModifyDate[2];//最近修改时间

uint8_t LowCluster[2];//文件起始簇号低16位

uint8_t FileSize[4];//文件长度

};

struct longfile {

uint8_t attributeOfFile;//属性字节
```

```
uint8_t unicodeOfFile1[10];//长文件名Unicode码

uint8_t longFileSymbol;//长文件名目录项标志

uint8_t SystemReserve;//系统保留

uint8_t checkNum;//校验值

uint8_t unicodeOfFile2[12];//长文件名Unicode码

uint8_t FileStartCluster[2];//文件起始簇号，常置为0

uint8_t unicodeOfFile3[4];//长文件名Unicode码

};


//这里不建议使用链表的格式，还不如使用栈，这样可以先进后出，刚好符合长文件的处理

struct longfile_list

{

struct longfile* lf;

struct longfile_list* prev;

struct longfile_list* next;

};

//rootdir（512字节）

struct rootdir {

shortfile shortfile[16];//因为每次只能读512字节

};

//fat表的基本信息

struct fatInfo {

uint32_t reserveSector;//保留扇区

uint32_t FatPerSector;//每个Fat表扇区数

uint32_t fat1;//fat1起始扇区

uint32_t fat2;//fat2起始扇区

uint32_t rootdir;//rootdir起始扇区

uint32_t SectorPercluster;//每个簇的扇区数
```

```
};

//簇号链

struct clusterChain {

uint32_t cluster[128];//每个clusterchain有128个目录项

};

struct fileInfo {

std::string fileName;//文件名字

uint32_t firstCluster;//首簇号

uint8_t fileClass;

};
```

# 宽字符处理

因为长文件是宽字符来处理，包括你的中文应该也是会记成2个字节来算。

得到三个unicode的长度

```
int len1 = sizeof(temp->lf->unicodeOfFile1) / sizeof(uint8_t);
int len2 = sizeof(temp->lf->unicodeOfFile2) / sizeof(uint8_t);
int len3 = sizeof(temp->lf->unicodeOfFile3) / sizeof(uint8_t);
```

然后进行字符拼接

```
memcpy(filename, temp->lf->unicodeOfFile1, len1);
memcpy(filename + len1, temp->lf->unicodeOfFile2, len2);
memcpy(filename + len1 + len2, temp->lf->unicodeOfFile3, len3);
```

然后因为是一个宽字符，所以你们需要将两个字节拼接放入一个宽字符进行处理

```
wchar_t tempWchar[MAX_PATH] = { '0' };
```

这样提取处理后，你们可以选择使用wcout去输出宽字符，也可以将宽字符转为 string ，当然我更推荐后

者。

# 可以用这个API去试着处理 WideCharToMultiByte ，转换成string。

# 提醒

在磁盘里面放文件，最好是先创建好的文件复制进去，因为如果你是新建文件再重命名，以前的 新建

文件.txt 这个文件还是会存在你的文件项里面。