编号：_____

实验	一	二	三	四	五	六	七	八	总评	教师签名
成绩										

武汉大学国家网络安全学院

课程实验(设计)报告

题　　目：软件安全实验

专业(班)：信息安全

学　　号：2021302181156

姓　　名：赵伯侯

课程名称：软件安全实验

任课教师：赵磊

2023 年 10 月 15 日

目 录

实验 1　磁盘格式与数据恢复

1.1 实验名称

磁盘格式与数据恢复

1.2 实验目的

1) 了解磁盘的物理和逻辑结构
2) 熟悉 FAT32 文件系统
3) 学会使用磁盘编辑软件
4) 了解文件删除、格式化的基本原理
5) 能够利用工具或者手工恢复被删除的文件

1.3 实验步骤及内容

1.3 实验步骤及内容

第一阶段：

- 熟悉 WinHex 的使用。
  - 熟悉磁盘工具的使用。
  - 利用 WinHex 查看物理磁盘和逻辑磁盘。
  - 了解 WinHex 中相关工具的用法。

第二阶段：

- 分析本地硬盘的主引导扇区
- 利用磁盘编辑工具查看 MBR 磁盘分区并分析：
  - 主引导扇区由哪些部分组成？
  - 四个主分区项的内容各代表什么？
  - 分析主扩展分区表的结构。
  - 通过分区项来确定每个本地逻辑盘的位置以及大小，并画出本地硬盘的逻辑结构。
    - 每个本地盘的开始扇区位置，总扇区数，结束扇区位置，各扩展分区表扇区位置，保留空间数量。
- 利用磁盘编辑工具查看 GPT 磁盘分区并分析
  - GPT 分区结构与 MBR 的具体差异有哪些？
  - 主分区头所在扇区包括哪些重要内容，验证这些重要内容的有效性。
  - 通过分区节点分析自己硬盘的各分区信息。

第三阶段：

- 熟悉 FAT32 文件格式。
  - 用 WinHex 打开某个 FAT32 分区格式的逻辑盘。
  - 查看该逻辑盘的起始扇区，分析起始扇区中的相关字段（BPB:BIOS Parameter Block）。
  - 查看 FAT1 和 FAT2 的内容和大小。
  - 查看该逻辑盘的根目录区。
  - 查看某个文件的目录项结构和 FAT 链以及具体存储位置。
  - 在根目录下建立文本文件：test-学号后 3 位.txt，其中填充 60K 左右的文本字符保存(注意:先行存储其他数据使得该文件的首簇高位不为 0)。
  - 查看该文件的目录项，对其进行分析，并得到该文件所在位置以及大小。
  - 查看首簇位置，并得到簇链表。通过簇链表查看该文件内容。
  - 记录首簇位置（14H-15H,1AH-1BH）和文件大小（1CH-1FH）。

第四阶段：

- 手工恢复被删除的文件
  - 删除前面所建立的文件。(del&shift+del)
  - 利用 WinHex 在该文件所在盘符查找该.txt 文件的目录项。
  - 查看目录项的变化。
  - 利用该残余目录项来计算被删除的文件所在的位置。
  - 手工恢复该文件（文件名、首簇高位、簇链表修复）。

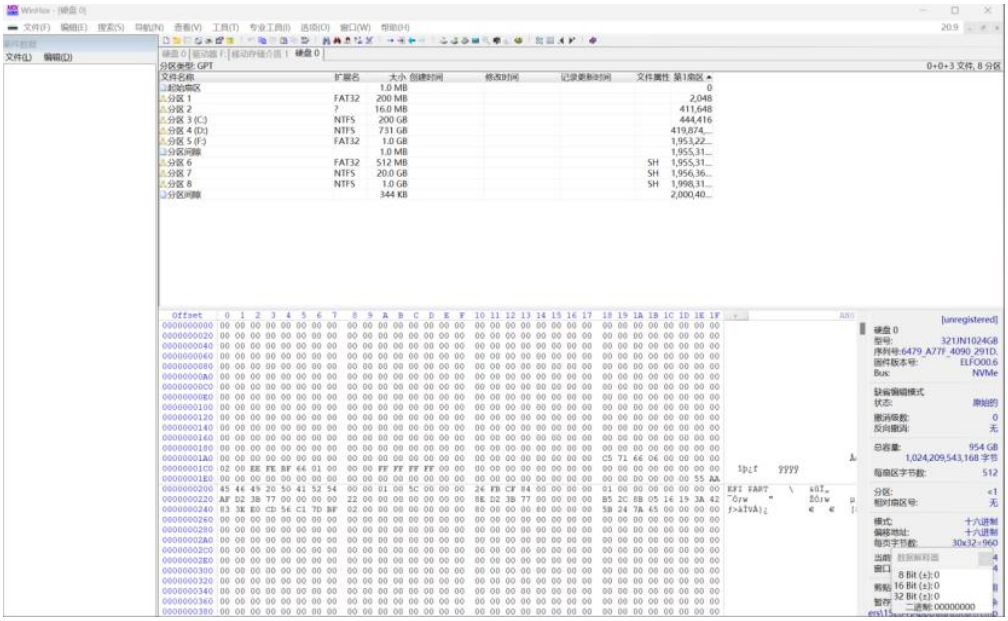
课后习题思考：

- 在磁盘分区过程中，用户提供了哪些信息？分析分区工具的工作原理。
- 高级格式化与低级格式化的具体原理和区别是什么？
- 查找资料，对 NTFS 分区的总体结构进行分析，尝试对 NTFS 下删除的文件进行手工恢复。
- 用数据粉碎工具（如金山、360、Strongdisk 等）粉碎指定文件，分析其数据粉碎原理。
- 通过分区表看到的分区字节数为何与资源管理器中看到的分区字节数有差异？
- 如果删除的文件是长文件名，如何恢复所有文件名。

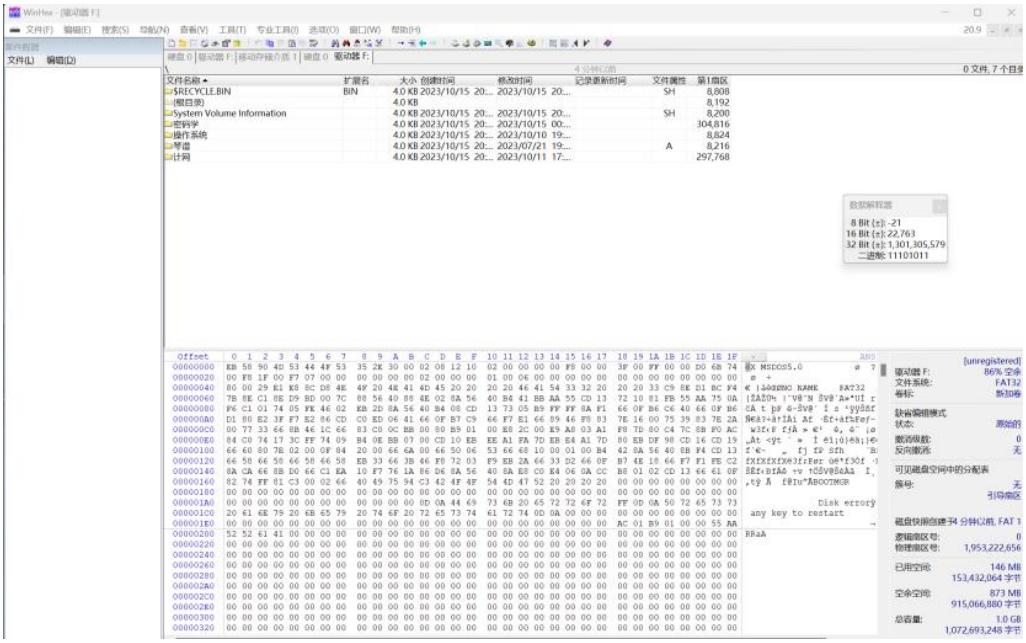
1.4 实验关键过程、数据及其分析

1.4.1 利用 WinHex 查看物理磁盘和逻辑磁盘。

利用 WinHex 打开本地的物理盘的情况如下图所示，存在有 8 个分区和 2 个分区间隔。该硬盘的分区类型为 GPT 类型，其中各个分区有 FAT32 类型的分区，也有 NTFS 类型的分区。

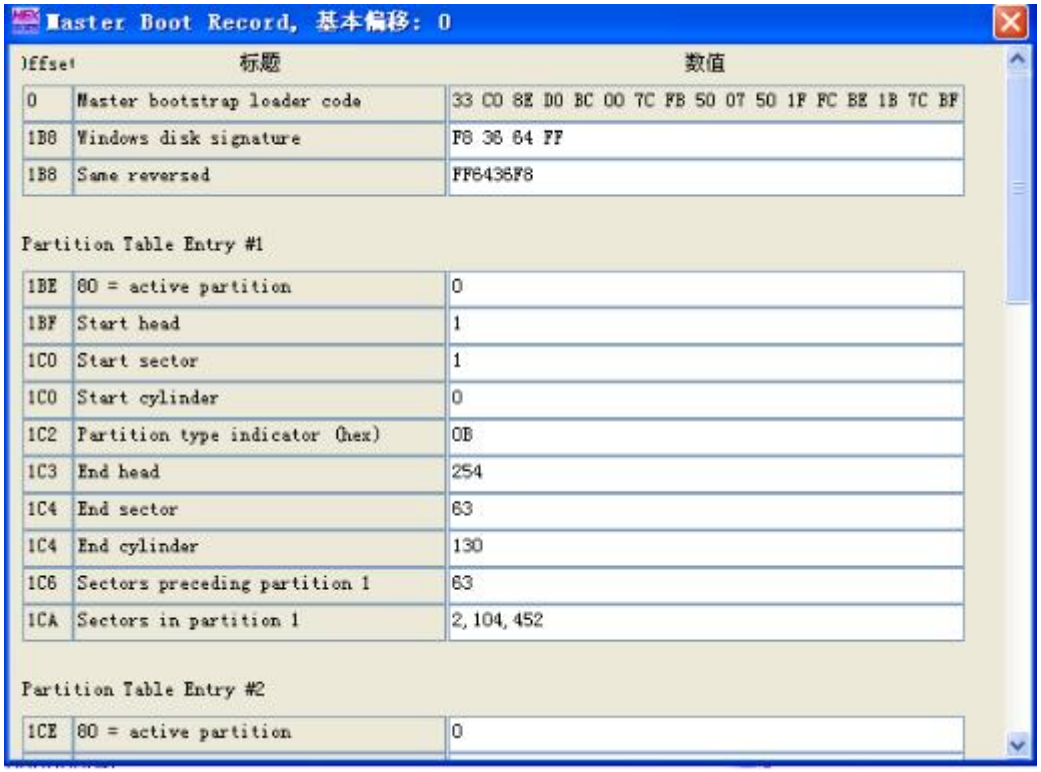


使用 WinHex 打开本地的逻辑分区后可以看到在逻辑分区中的众多文件目录。还可以看到该分区的文件系统是 FAT32 格式。



1.4.2 分析主引导扇区的组成部分。

将主引导扇区的分区表模板打开后如下图所示。



Master Boot Record, 基本偏移: 0

Offset	标题	数值
0	Master bootstrap loader code	33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C BF
1B8	Windows disk signature	F8 36 64 FF
1B8	Same reversed	FF6436F8

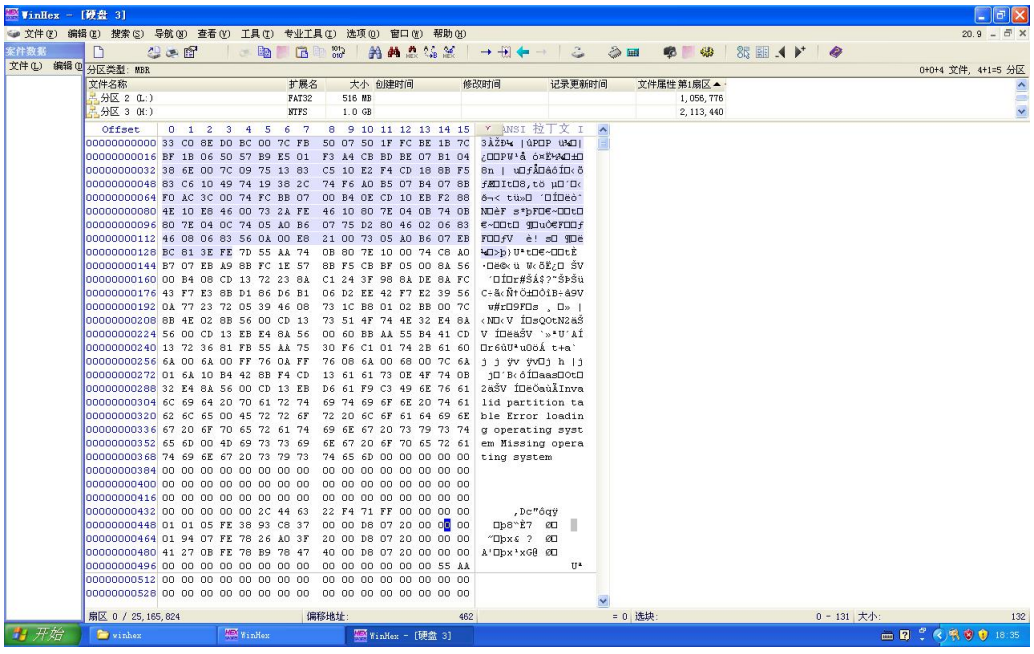
Partition Table Entry #1

1BE	80 = active partition	0
1BF	Start head	1
1C0	Start sector	1
1C0	Start cylinder	0
1C2	Partition type indicator (hex)	0B
1C3	End head	254
1C4	End sector	63
1C4	End cylinder	130
1C6	Sectors preceding partition 1	63
1CA	Sectors in partition 1	2,104,452

Partition Table Entry #2

1CE	80 = active partition	0

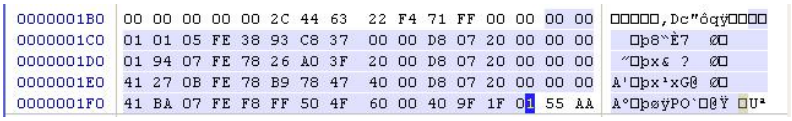
分区表中的具体内容如下图所示



在一个主引导扇区中，前 1B8H（440）个字节为主引导扇区的引导代码，从第 1BEH（446）个字节开始为主引导扇区的分区表，分区表占用 64 个字节，分区表中包含有四个分区项的具体内容，最后主引导扇区以 55AA 结尾。

1.4.3 分析四个主分区项的内容

在本次实验中使用的物理盘的四个主分区项的内容如下图所示，

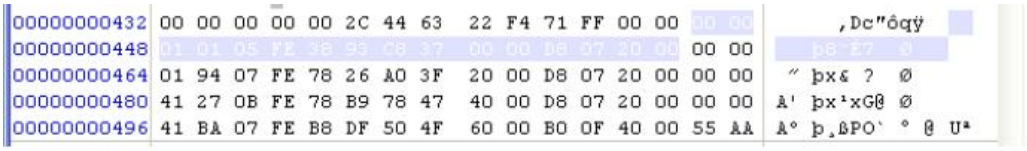


其中分区表由 64B 组成，分区表一共有四个，其中每个分区表的结构如下表所示。

存储字节	数据内容	分区 1	分区 2	分区 3	分区 4
1	引导标志	00	00	00	00
2	本分区的起始磁道号	00	00	00	00
3	本分区的起始扇区号	01	01	41	41
4	本分区的起始柱面号	01	94	27	BA
5	分区类型	05	07	0B	07
6	本分区的结束磁道号	FE	FE	FE	FE
7	本分区的结束扇区号	38	78	78	F8
8	本分区的结束柱面号	93	26	B9	FF
9-12	本分区之前已经占用的扇区数	000037C8	00203FA0	00404778	00604F50
13-16	本分区的总扇区数	002007D8	002007D8	002007D8	011F9F40

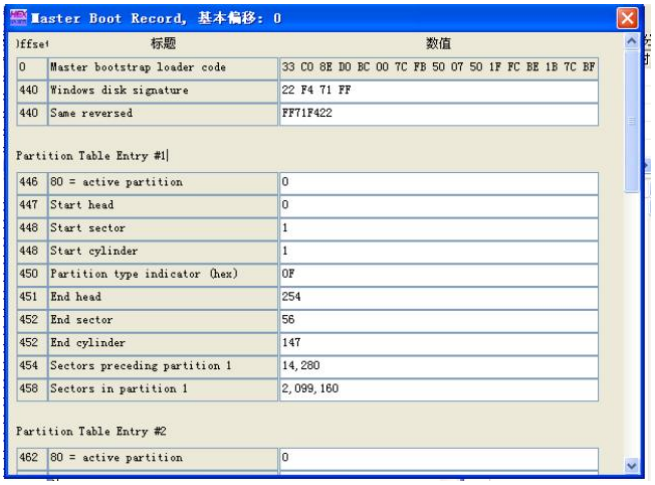
1.4.4 分析主扩展分区表的结构。

打开一个主拓展分区的分区表如下图所示。



00000000432	00	00	00	00	00	2C	44	63	22	F4	71	FF	00	00	,Dc"ôqÿ
00000000448	01	01	05	FE	38	93	C8	37	00	00	D8	07	20	00	00	00	þ8 È7 Ø
00000000464	01	94	07	FE	78	26	A0	3F	20	00	D8	07	20	00	00	00	″ þx& ? Ø
00000000480	41	27	0B	FE	78	B9	78	47	40	00	D8	07	20	00	00	00	A' þx¹xG@ Ø
00000000496	41	BA	07	FE	B8	DF	50	4F	60	00	B0	0F	40	00	55	AA	Aº þ¸ßPO` ° @ Uª

使用 Winhex 打开其模板如下图所示。



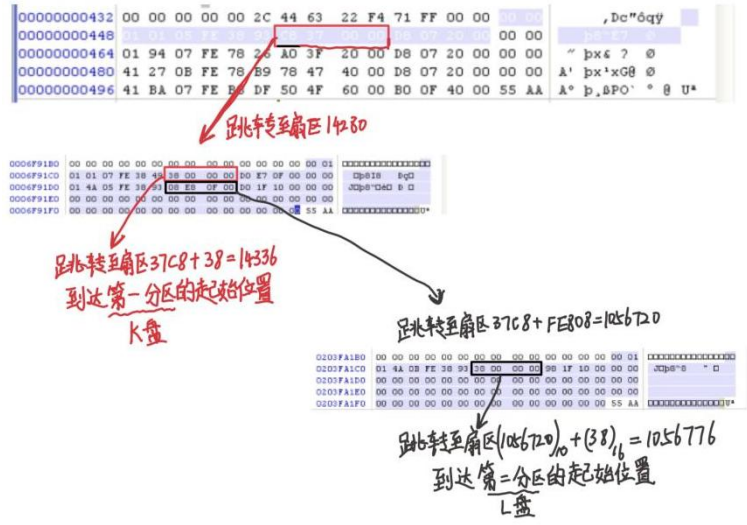
Master Boot Record, 基本偏移: 0

Offset	标题	数值
0	Master bootstrap loader code	33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C BF
440	Windows disk signature	22 F4 71 FF
440	Same reversed	FF71F422
	Partition Table Entry #1	
446	80 = active partition	0
447	Start head	0
448	Start sector	1
448	Start cylinder	1
450	Partition type indicator (hex)	0F
451	End head	254
452	End sector	56
452	End cylinder	147
454	Sectors preceding partition 1	14,280
458	Sectors in partition 1	2,099,160
	Partition Table Entry #2	
462	80 = active partition	0

一个拓展分区中可以有多个逻辑驱动器，扩展分区中的每个逻辑驱动器的分区信息都存在一个类似于 MBR 的扩展引导记录中，如上图所示，扩展引导记录包括分区表和结束标志“55 AA”，没有引导代码部分。

EBR 中分区表的第一项描述第一个逻辑驱动器，第二项指向下一个逻辑驱动器的 EBR。如果不存在下一个逻辑驱动器，第二项就不需要使用。下图为在磁盘管理器中查看逻辑磁盘状态。绿色框中的部分就是主拓展分区。

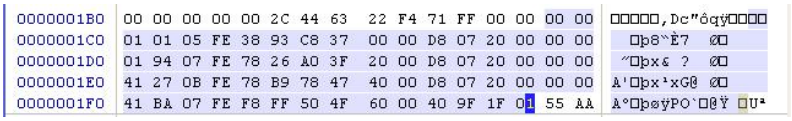
根据主拓展分区的分区表绘制出的主拓展分区在硬盘中的结构如下图所示。



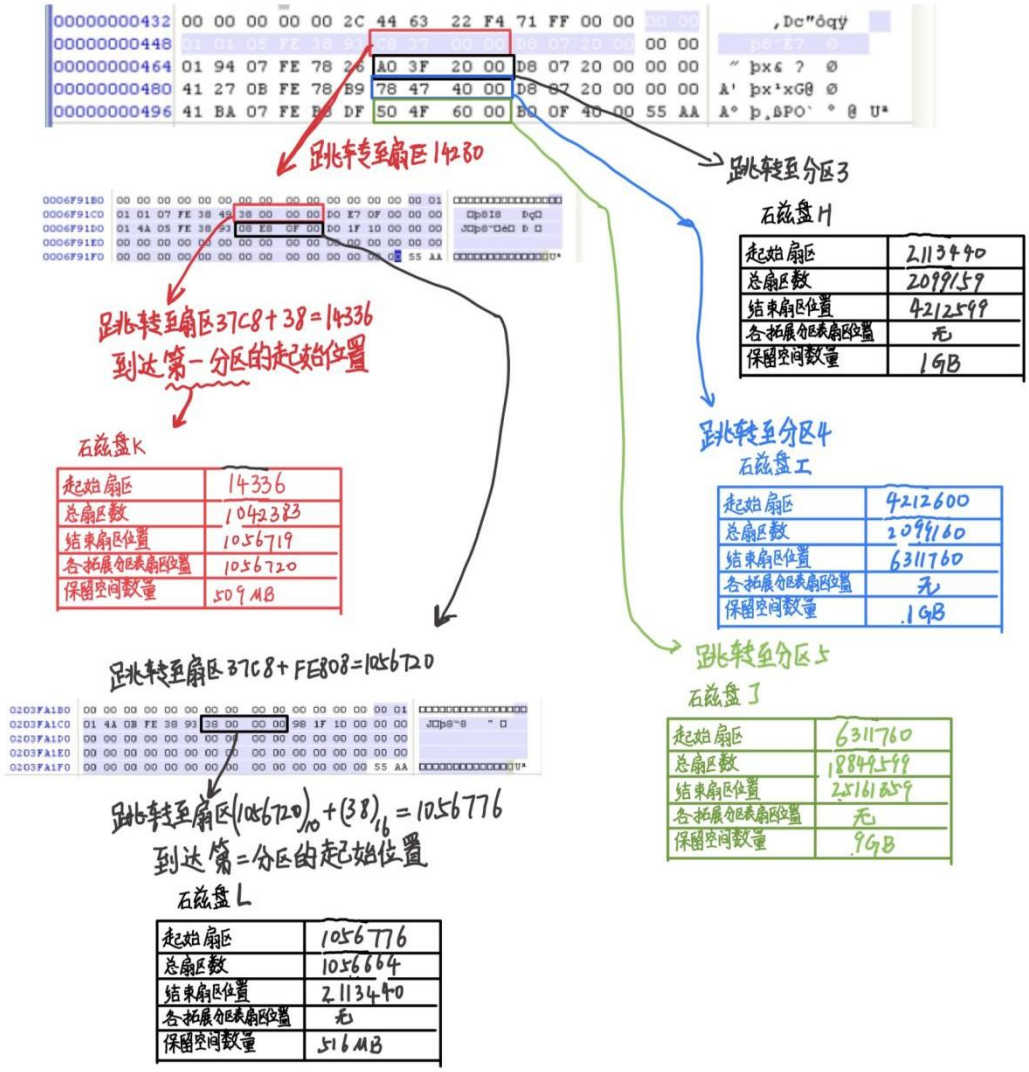
1.4.5 分析每个本地逻辑盘的位置大小和逻辑结构及其扇区信息。

在本次实验中分析磁盘 3 每个逻辑盘的位置大小和逻辑结构，首先在磁盘管理器中查看逻辑盘结构如下图所示。

打开其主分区表如下所示

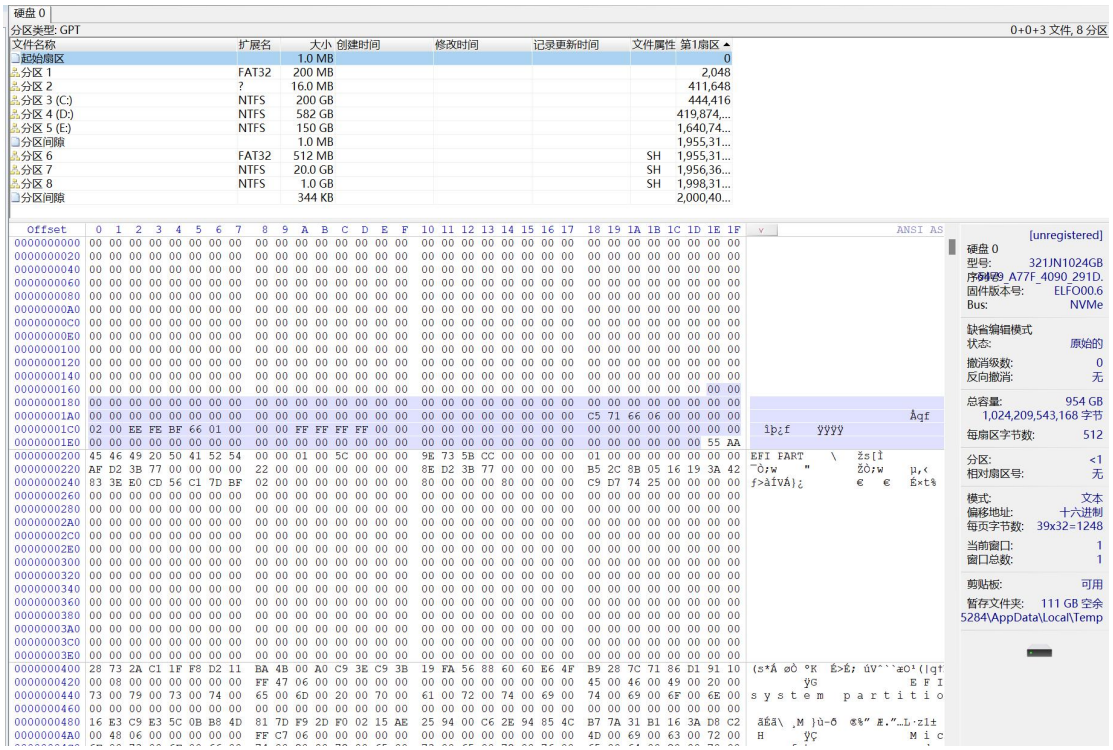


根据分区表绘制出的每一个逻辑盘的信息如下图所示。

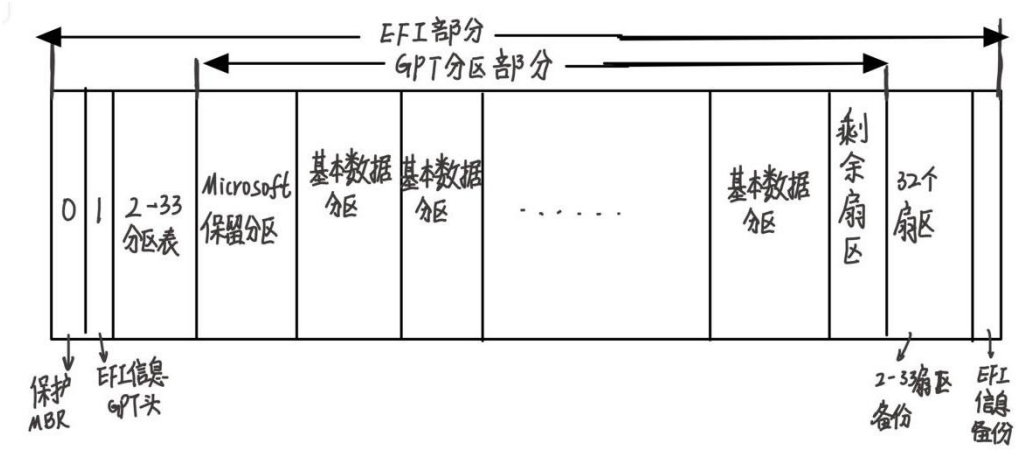


1.4.6 分析 GPT 分区的格式。

使用 WinHex 在物理机上打开一个硬盘后硬盘内容如下图所示显示分区类型为 GPT 类型。



GPT 分区的整体结构如下图所示。



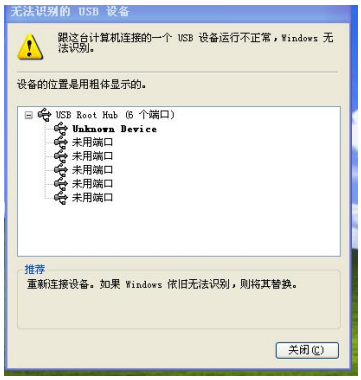
1.4.7GPT 分区结构与 MBR 的具体差异有哪些？

1.MBR 中分区表的组织将磁盘的可寻址存储空间限制为 2TB（232×512 字节）。
2.MBR 只支持 4 个主分区，或者 3 个主分区和 1 个扩展分区的组合。如果要创建更多分区，则需要将其中一个主分区设为“扩展分区”，然后在扩展分区内创建更多逻辑分区。否则，磁盘将转换为动态磁盘。
3.在 MBR 中，分区表的大小是固定的；在 GPT 分区表头中可自定义分区数量的最大值，也就是说 GPT 分区表的大小不是固定的。

4. GPT 磁盘具有冗余的主分区表和备份分区表，可以优化分区数据结构的完整性。
通常来说，MBR 和 BIOS（MBR+BIOS）、GPT 和 UEFI（GPT+UEFI）是相辅相成的。这对于某些操作系统（例如 Windows）是强制性的，但是对于其他操作系统（例如 Linux）来说是可以选择的。

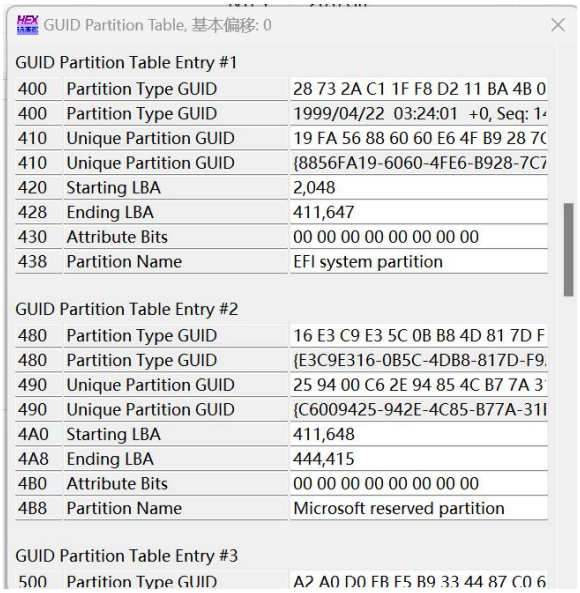
1.4.8 主分区头所在扇区包括哪些重要内容，验证这些重要内容的有效性。

主分区头所在的扇区有一个标识为 0xEE 的分区，以此来表示这块硬盘使用 GPT 分区表。不能识别 GPT 硬盘的操作系统通常会识别出一个未知类型的分区，并且拒绝对硬盘进行操作，除非用户特别要求删除这个分区。这就避免了意外删除分区的危险。另外，能够识别 GPT 分区表的操作系统会检查保护 MBR 中的分区表，如果分区类型不是 0xEE 或者 MBR 分区表中有多个项，也会拒绝对硬盘进行操作。
验证该内容的有效性。找到一个 GPT 分区类型的 U 盘之后将其连接到 WinXP 操作系统中，WinXP 能够识别 MBR 类型的磁盘，连接好后会显示如下错误。由此验证 GPT 分区头所在的扇区能够拒绝不能识别 GPT 分区的操作系统对其进行读取操作。



1.4.9 通过分区节点分析自己硬盘的各分区信息。

打开 GPT 分区的磁盘的分区表后如下图所示。

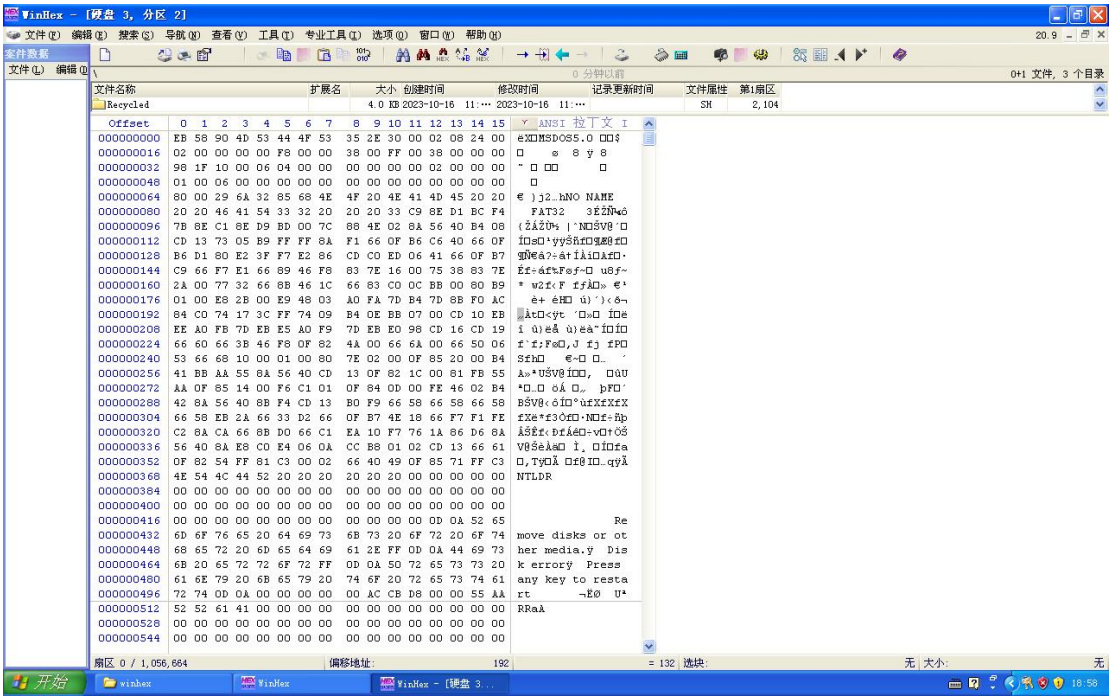


以下分析第一分区和第二分区的信息如下表所示

内容	分区 1	分区 2
分区类型 GUID	28 73 2A C1 1F F8 D2 11 BA 4B 00 A0 C9 3E C9 3B	16 E3 C9 E3 5C 0B B8 4D 81 7D F9 2D F0 02 15 AE
分区 GUID	19 FA 56 88 60 60 E6 4F B9 28 7C 71 86 D1 91 10	25 94 00 C6 2E 94 85 4C B7 7A 31 B1 16 3A D8 C2
分区起始地址	2,048	411,648
分区结束地址	411,647	444,415
分区属性	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
分区名	EFI system partition	Microsoft reserved partition

1.4.10 分析 FAT32 分区格式的逻辑盘的起始扇区。

用 WinHex 在虚拟机中打开 FAT32 分区格式的逻辑盘 I 的起始扇区如下图所示。



打开其 BIOS Paraneter Block 如下图所示，

Boot Sector FAT32, 基本偏移: 0

Offset	标题	数值
0	JMP instruction	EB 58 90
3	OEM	MSDOS5.0

BIOS Paraneter Block

B	Bytes per sector	512
D	Sectors per cluster	8
E	Reserved sectors	38
10	Number of FATs	2
11	Root entries (unused)	0
13	Sectors (on small volu	0
15	Media descriptor (hex)	F8
16	Sectors per FAT (small	0
18	Sectors per track	58
1A	Heads	255
1C	Hidden sectors	4,212,600
20	Sectors (on large volu	2,099,160

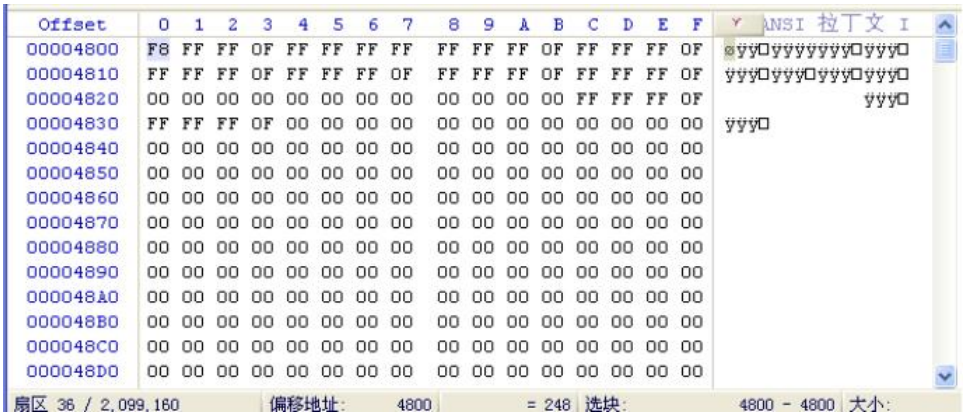
FAT32 Section

分析其 BIOS Paraneter Block 的组成和在该逻辑盘中的值如下表所示。

偏移量	字节数	含义	值
0x00B	2	每扇区字数	0x0200
0x00D	1	每簇扇区数	0x08
0x00E	2	保留扇区数	0x0024
0x010	1	FAT 个数	0x02
0x011	2	根目录项数，FAT32 以突破该限制，无效	0x0000
0x013	2	扇区总数，小于 32M 使用	0x0000
0x015	1	存储介质描述	0x0F8
0x016	2	每 FAT 表占用扇区数 ，小于 32M 使用	0x0000
0x018	2	逻辑每磁道扇区数	0x0038
0x01A	2	逻辑磁头数	0x00FF
0x01C	4	系统隐含扇区数	0x00000038
0x020	4	扇区总数，大于 32M 使用	0x00101F98
0x024	4	每 FAT 表扇区数，大于 32M 使用	0x00000406
0x028	2	标记	0x0000
0x02A	2	版本 (通常为零)	0x0000
0x02C	4	根目录起始簇	0x00000002
0x030	2	Boot 占用扇区数	0x0001
0x032	2	备份引导扇区位置	0x0006
0x034	14	保留	0x00800000000000000000000000
0x042	1	扩展引导标记	0x29
0x043	4	序列号	0x6885326A
0x047	10	卷标	转成字符即“NO NAME”
0x052	8	文件系统	转成字符即“FAT32”

1.4.11 查看该逻辑盘的 FAT1 和 FAT2 的内容和大小。

打开 FAT1 和 FAT2 的内容如下图所示。其中 FAT2 的内容和 FAT1 相同，因为 FAT2 是 FAT 表的备份。其中大部分内容为簇占用标记 FFFFFF0F 表示该簇是一个文件的结束。其余为指向下一个簇的位置，在该磁盘中没有发现坏簇标记，即该磁盘中尚且不存在坏簇。



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00004800	F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	FF	FF	FF	0F
00004810	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F
00004820	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	FF	0F
00004830	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00004840	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004850	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004860	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004870	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004880	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004890	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000048A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000048B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000048C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000048D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

扇区 36 / 2,099,160　偏移地址: 4800　= 248　选块: 4800 - 4800　大小:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00104400	F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	FF	FF	FF	0F
00104410	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F
00104420	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	FF	0F
00104430	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00104440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00104450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00104460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00104470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00104480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00104490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001044A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001044B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001044C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001044D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

扇区 2,082 / 2,099,160　偏移地址: 104400　= 248　选块: 4800 - 4800　大小:

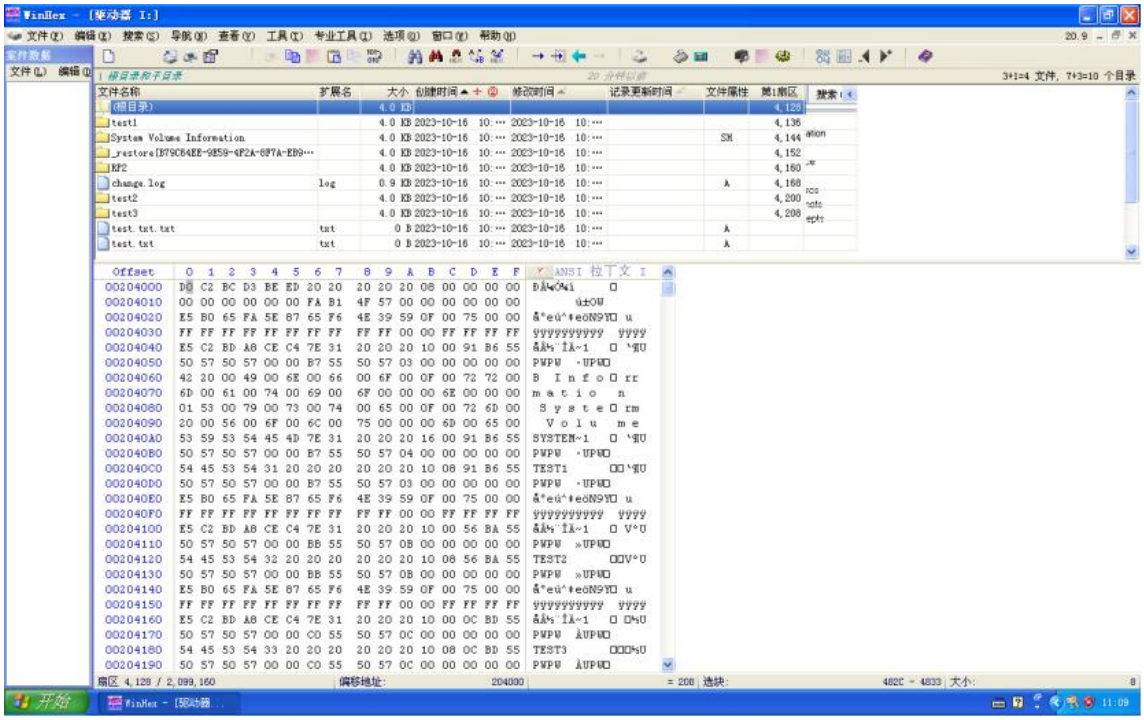
该逻辑盘中一个扇区占 512 字节，一簇 8 个扇区

于是一簇 8*512=4kb

F 盘 1GB，计算 1*1024*1024/4 得簇数共有 262144 个，一簇在 FAT 表中占 4 字节，故计算得 FAT 表大小应为 4*262144B=1MB。

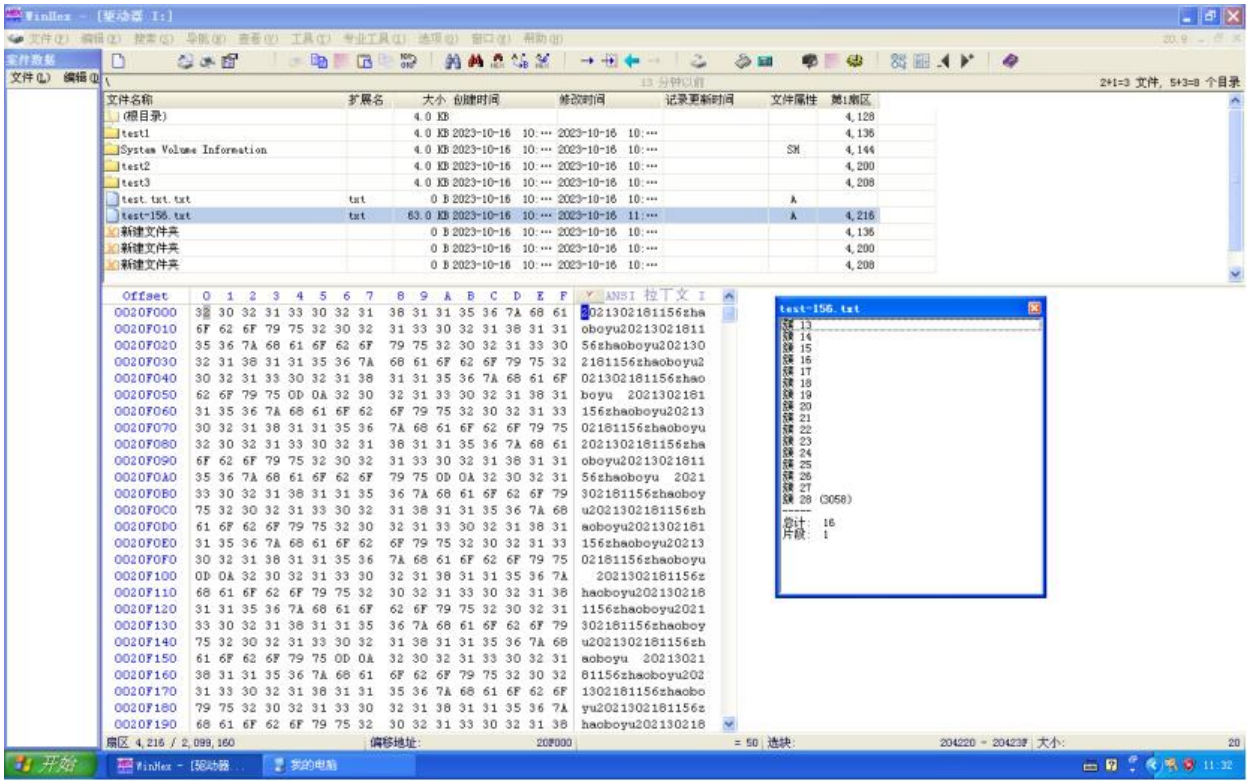
1.4.12 查看该逻辑盘的根目录区。

该逻辑盘的根目录去如下图所示。

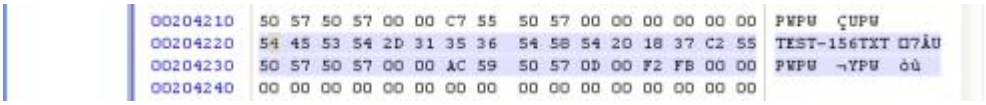


1.4.13 新建文件并分析其目录项

在 test-156.txt 文件中存有 63KB 的数据后观察其簇的结构，如下图左下角显示的是该文件的具体存储位置，右下角显示出的是该文件的簇的格式。

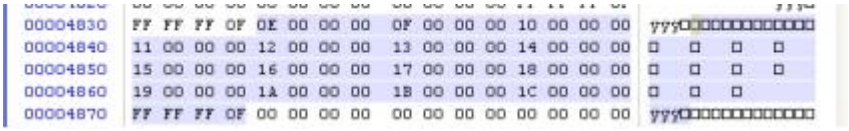


得到的该文件的目录项如下所示。00-07 代表文件的拓展名，08-0A 代表文件的拓展名，14H-15H 为起始簇号高 16 位（0000）。1AH-1BH 为起始簇号低十六位（0D00）。1CH-1FH 为文件字节长度（0000FBF2）,为 64498 字节。起始簇号 0000000DH，为 13 号。



1.4.14 分析新建文件的簇相关信息

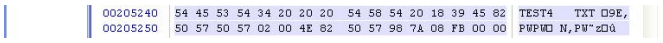
分析得簇列表如下所示，该文件共占有 16 个簇，因此其簇列表有着 16*4=64B 的内容，其首簇数据为 0000000E=14 指向下一个簇号。之后的每一个簇内容都指向下一个簇的位置。最后一个簇的内容为 FFFFFF0F



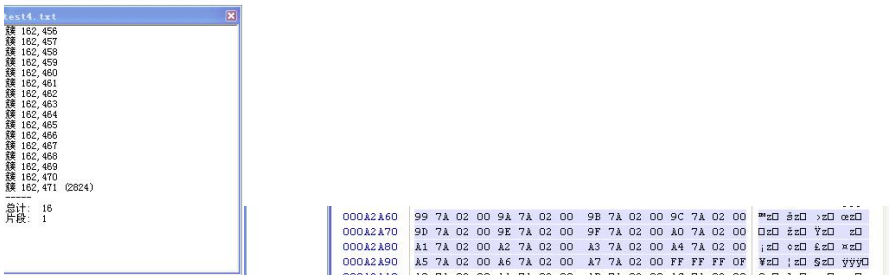
1.4.15 删除文件的恢复

（1）恢复前文件信息。

在删除文件之前首先得到该文件的目录表项的内容如下图所示，观察该文件的信息我们可以看到，该文件的簇高位为 0002。文件名为 test4。



查看该文件的簇列表如下图所示，该文件一共存储在 16 个簇中，粗链表在内存中保存在 A2A60-A2A90 区域内。



（2）恢复后文件信息。

将文件使用 shift+del 删除之后查看文件的目录项发现簇高位由原来的 0002 被清零，簇列表被全部清零。但在文件目录项中仍然能够查看到文件其他位置的目录项。

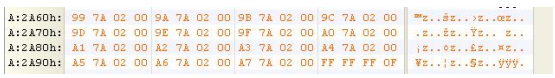
（3）对 test4 进行数据恢复操作

首先恢复文件目录项，打开 test4 之前和之后的文件如下图所示，观察其簇高位的数值。

由此推测出该文件的簇高位也应该为 02，在 010Editor 中修改簇高位的值，将其该为我们的推测值 02。

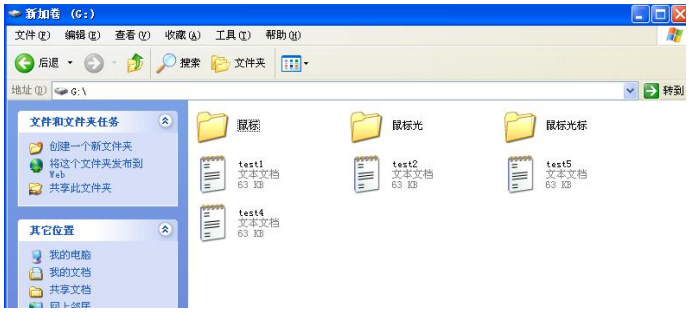
然后观察文件目录项可以发现文件的头由原来的 54 被改为了 E5，在此我们将其改回，修改完成的文件目录项如下图所示。

接下来修改簇列表，已知簇高位为 02，簇低位为 7A 98，由此，我们将簇低位加 1 后写入第一个簇的前两个字节中去，然后写入簇高位的一个字节，在最后一个字节补上 00，由此修改完第一个簇，将后面的簇的簇地位依次加 1 后填入簇链表中。将簇列表修改完成后的结果如下图所示，



（4）验证结果

重新打开删除文件 test4 所对应的盘区发现被删除的文件已经被恢复。



1.4.16 课后习题思考

（1）在磁盘分区过程中，用户提供了哪些信息？分析分区工具的工作原理。

用户需要提供的信息仅有新分区的文件习统和新分区的空间大小。

工作原理就是首先分区工具读取硬盘上的分区表，获取该磁盘中已知的分区结构和信息，然后根据用户输入的参数计算出新分区的起始位置和大小，并在分区表中创建相应的分区项，同时在磁盘上为新分区分配空间。

然后根据用户提供的文件系统对新分区进行格式化，在新分区的位置创建新的文件系统结构，存储元数据，并准备分区以存储文件和目录。

最后分区工具将对分区表进行更新，将新分区的信息写入分区表中。另外如果是在主拓展分区中还需要创建对应的 EBR 表项。

（2）高级格式化与低级格式化的具体原理和区别是什么？

原理：

低级格式化是指对物理磁盘介质进行处理，重新建立磁道和扇区的布局。在低级格式化过程中，磁盘驱动器会对磁盘表面进行物理处理，包括将磁道划分为扇区、标定磁道和扇区的位置以及校准磁头等操作。低级格式化通常由磁盘驱动器的制造商在生产过程中完成，用户很少需要手动进行低级格式化。

高级格式化是指在低级格式化之后对磁盘进行逻辑上的处理，包括创建文件系统、分配文件系统结构和元数据等。高级格式化通常由操作系统或格式化工具提供，用户可以在使用新磁盘或重新格式化磁盘时执行高级格式化。

区别：

在原理层面，低级格式化处理物理磁盘介质，重新建立磁道和扇区的布局；而高级格式化处理逻辑上的文件系统结构和元数据。

在作用上，低级格式化主要用于磁盘驱动器的生产阶段，由制造商完成，目的是初始化磁盘；高级格式化用于创建文件系统、分区和准备磁盘以存储数据。

在可控性上，低级格式化通常由磁盘驱动器制造商进行，用户很少需要手动进行；而高级格式化由用户或操作系统控制，并提供了更多的选项和灵活性，例如选择文件系统类型、分区大小等。

（3）查找资料，对 NTFS 分区的总体结构进行分析，尝试对 NTFS 下删除的文件进行手工恢复。

NTFS 分区的总体结构包括以下几个部分。

主引导记录。NTFS 分区通常位于硬盘的主引导记录中。MBR 包含引导代码和分区表，用于引导操作系统和标识磁盘上的分区。

NTFS 引导扇区。NTFS 分区的第一个扇区是 NTFS 引导扇区（NTFS Boot Sector），也称为扇区 0。它包含了引导代码和关键的文件系统参数，如 OEM 标识、扇区大小、分区偏移等。

文件分配表。NTFS 使用了一种称为文件分配表（File Allocation Table）或 MFT（Master File Table）的结构来管理文件和目录的分配和存储。MFT 是 NTFS 中的核心组成部分，记录了文件和目录的属性、大小、权限等信息。

文件记录。NTFS 中的每个文件和目录都在 MFT 中有一个相应的文件记录。文件记录包含了文件的属性、文件名、数据的物理位置等信息。对于较小的文件，文件记录直接存储在 MFT 中；对于较大的文件，文件记录中存储了指向数据存储位置的指针。

数据存储区。NTFS 使用数据存储区来存储文件和目录的实际数据。对于较小的文件，数据存储区可以直接包含文件数据；对于较大的文件，数据存储区中存储了数据的扇区位置。

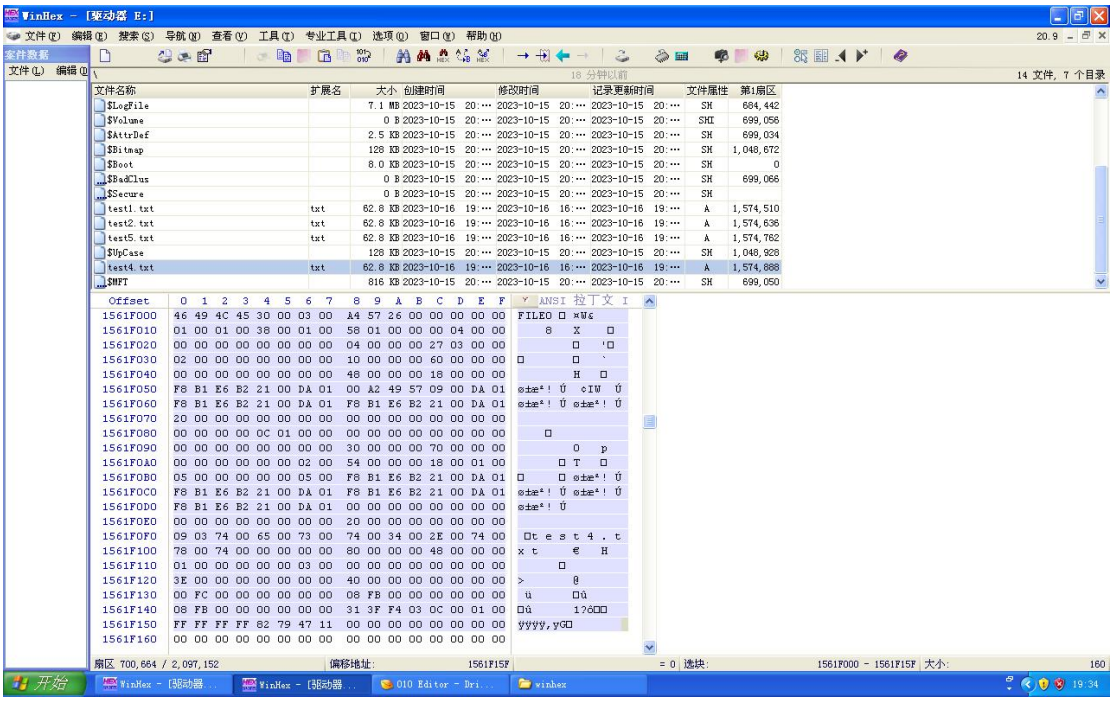
文件属性。NTFS 支持各种文件属性，包括标准属性（如文件大小、创建时间、修改时间等）和扩展属性（如压缩属性、加密属性等）。这些属性存储在文件记录中，并提供了对文件的详细描述和管理。

安全描述符。NTFS 使用安全描述符来管理文件和目录的访问权限。安全描述符包含了访问控制列表（Access Control List, ACL），用于定义允许或拒绝访问文件的用户和组。

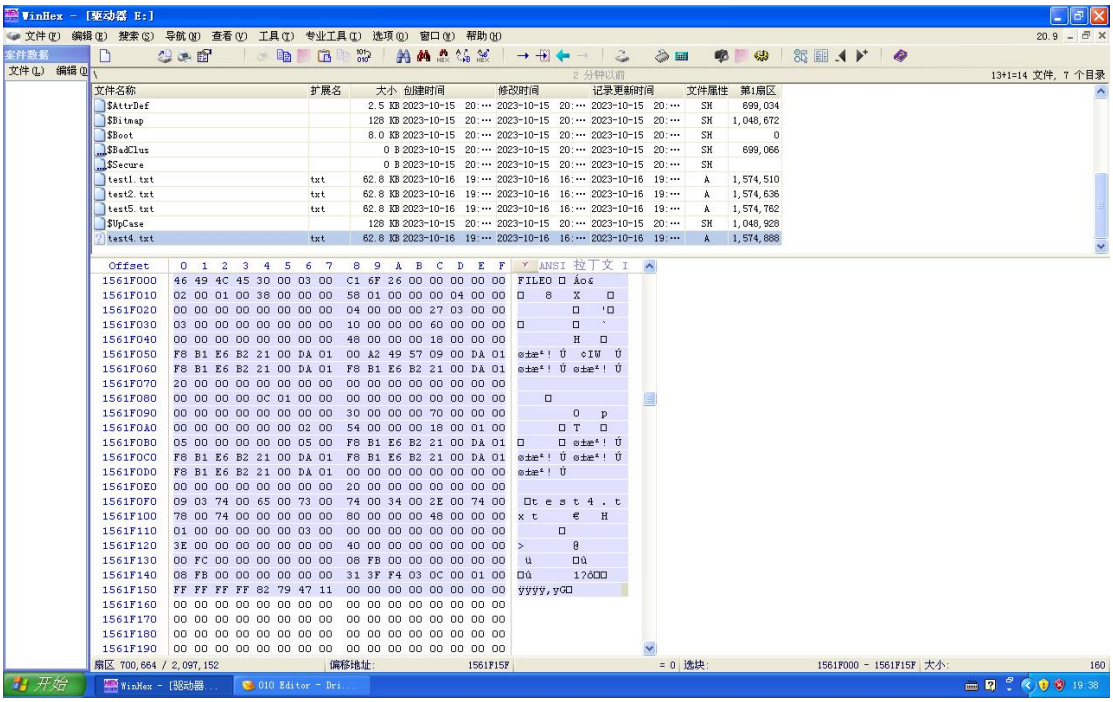
日志文件。NTFS 使用日志文件（Log File）来记录对文件系统的更改操作。日志文件可以用于恢复文件系统的一致性，并提供了对系统崩溃或意外断电的保护。

以下对 NTFS 文件系统中文件进行数据恢复。

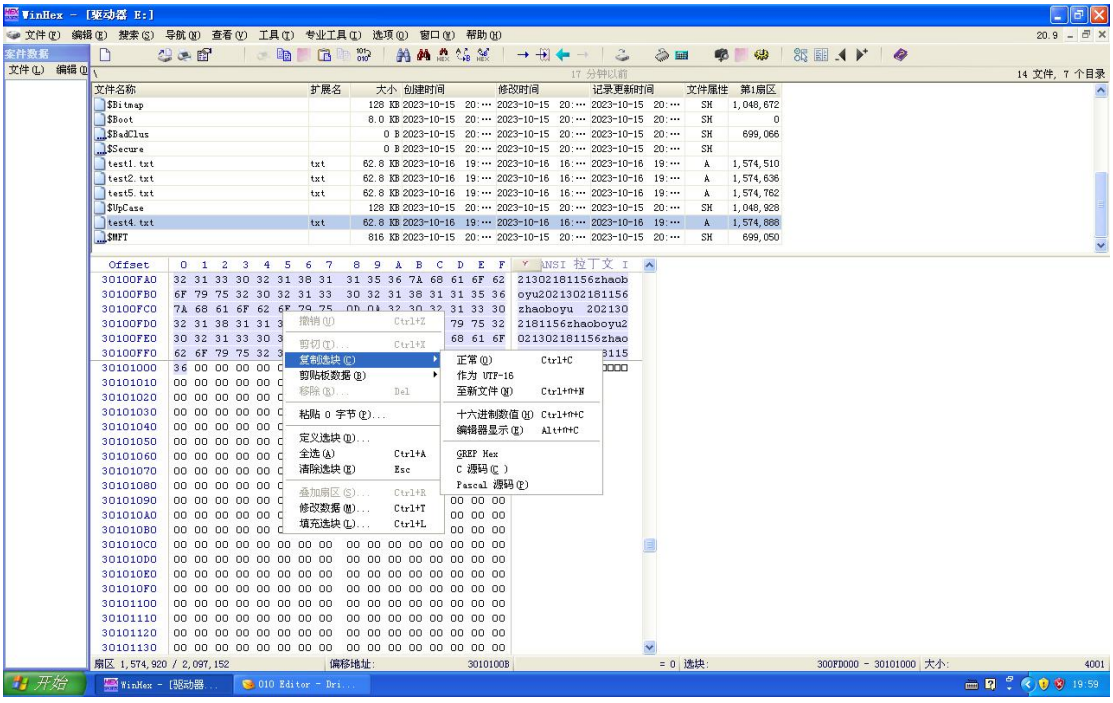
在删除前 test4 的文件目录如下图所示。



将 test4 彻底删除后其目录表如下图所示。

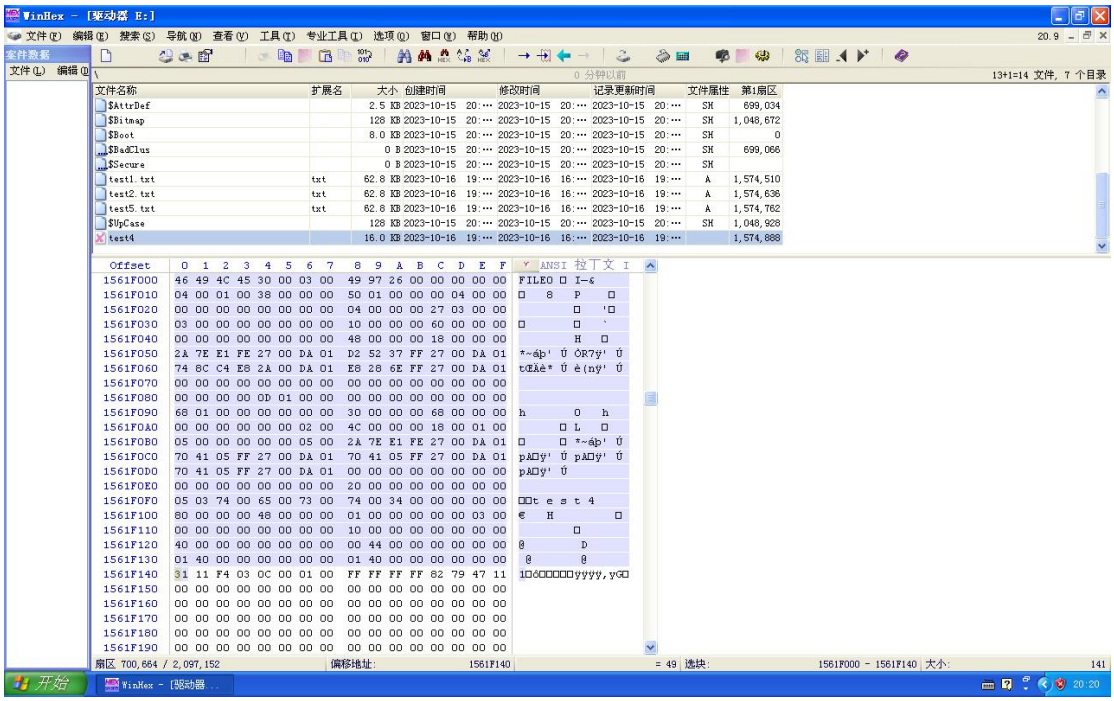


对目录表进行分析，在 80H 中保存有文件的内容，其中 01 代表这是非常驻属性，找到文件的簇流信息为 31 3F F4 03 0C　可得起始簇位置为 0C03F4H=787444,簇的大小为 10H=16，使用 winhex 跳转到起始簇 787444 的位置，并将其选择为块起始位置，尾部位置为 787444+16=787460，将选块复制进新文件即可完成数据恢复。



（4）用数据粉碎工具粉碎指定文件，分析其数据粉碎原理。

使用数据粉碎工具对 test4 进行删除后的结果如下图所示



在 test4 的原文件目录下的所有数据都已经被重写为其他数据，由此推断数据粉碎工具使用一种或多种覆盖算法来重复写入特定的数据模式到要删除的文件或磁盘空间上。这些算法通常使用随机数据、零值或者特定模式的数据进行覆盖。

（5）通过分区表看到的分区字节数为何与资源管理器中看到的分区字节数有差异？

在分区表中，分区字节数通常是通过读取分区表中的元数据来计算得出的。分区表中的元数据包含了有关分区的信息，如起始位置、结束位置等。通过计算这些信息，可以得到分区的字节数。

而资源管理器中看到的分区字节数则是通过读取文件系统的信息来计算得出的。文件系统包含了文件和目录的组织结构以及其在磁盘上的分布情况。资源管理器通过读取文件系统的信息，包括文件分配表和目录结构等，计算得出每个分区的字节数。

（6）如果删除的文件是长文件名，如何恢复所有文件名。

如果删除的是长文件名则可以通过查阅文件系统日志来进行数据恢复，某些文件系统会记录文件系统操作的日志，包括文件的创建、修改和删除等操作。通过分析文件系统日志，可以查找已删除文件的相关信息，包括文件名和位置。

或者可以借助文件标识符，在某些文件系统中，文件名和文件数据是分开存储的。文件名通常存储在目录中的文件标识符或类似的数据结构中，而文件数据则存储在不同的位置。通过查找文件标识符，可以找到已删除文件的元数据，包括文件名和其他属性。

1.5 实验体会和拓展思考

1.在我们平时使用电子设备时的删除功能可能并不是我们理解的删除，更像是将文件的标签进行了删除，而文件的本体还保存在硬盘中。
2.市面上所存在的众多文件恢复和文件彻底粉碎产品也是借用了这一原理，文件恢复产品通过恢复文件的标签或者说头目录表对文件本体进行恢复，而文件粉碎产品也是通过读取文件本体数据的各个位置将其覆盖为新的随机数据对文件进行改写。
3.数据文件在硬盘中的存放并不仅仅是简单的找到磁盘的空闲位置将文件整个放进去，而是将磁盘分成不同的区，不同的簇进行管理，文件所在的位置可能处在不同的簇中，这样做的目的是能够方便磁盘的管理，数据的存放以及数据的查找。
4.新分区类型的开发不仅要注重功能上的提升，同时还要注意能否与上一代设备兼容的问题，在 GPT 分区中的分区头部设立了一个保护分区来使得智能识别 MBR 分区类型的设备能够对其进行识别并报错，这一部分使得新开发的设备能够与老设备进行兼容。