# 实验四 设备管理

**[1]** **磁盘I/O API函数应用**

背景知识

**相关的API 介绍**

1.获取磁盘的基本信息的磁盘I/O API函数DeviceIoControl格式如下：

BOOL **DeviceIoControl**( HANDLE hDevice， 　 DWORD dwioControlCode，

　　　　　　　LPVOID lplnBuffer， 　 DWORD nlnBufferSize，

　　　　　　　LPVOID lpOutBuffer，　　 DWORD nOutBufferSize，

　　　　　　　LPDWORD lpBytesReturned，LPOVERLAPPED lpOverlapped　）；

．hDevice：所要进行操作的设备的句柄，它通过调用CreateFile函数来获得。

．dwIoControlCode：指定操作的控制代码。这个值用来辨别将要执行的指定的操作，以及对哪一种设备进行操作。对磁盘应设置为IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY。

．lpInBuffer：操作所要的输入数据缓冲区指针，NULL表示不需要输入数据。

．nInBufferSize：指定lpInBuffer所指向的缓冲区的大小（以字节为单位）。

．lpOutBuffer：接收操作输出的数据缓冲区指针，NULL表示操作没有产生输出数据。输出数据的缓冲区要足够大，对磁盘它采用固定的数据结构DISK\_GEOMETRY，格式如下：

struct DISK\_**GEOMETRY** {

unsigned bytesPerSector； unsigned sectorsPerTrack；

unsigned heads； unsigned cylinders； }

．nOutBufferSize：指定lpOutBuffer所指向的缓冲区的大小（以字节为单位）。

．lpBytesReturned：指向一个变量，它接收lpOutBuffer所指的缓冲区储存的数据个数。

．lpOverlapped：指向一个OVERLAPPED结构。

返回值：如果函数调用成功，返回值是一个非0值。如果函数调用失败GetLastError函数来获得相关的错误信息。

2.建立文件或打开一个已存在文件API函数CreateFile

该函数用来创建或打开下列对象（文件、管道、目录、邮件插口、控制台、通信资源、磁盘设备等）并返回一个用于读取该对象的句柄。

HANDLE **CreateFile** ( LPCTSTR lpFilename , DWORD dwDesiredAccess ，

DWORD dwShareMode ，LPSECURITY\_ATTRIBUTES lpSecurityAttributes ，

DWORD dwCreationDisposition , DWORD dwFlagsAndAttributes ，

HANDLE hTemplateFile )；

．lpFileName：指向一个以NULL结束的字符串的指针，该字符串用于创建或打开对象、指定对象名。

．dwDesiredAccess：指定对对象的访问类型，一个应用程序可以得到读、写、读写或设备查询访问等类型，此参数可以为下列值的任意一个组合值：

．0：指定对象的查询访问权限，一个应用程序可以不通过访问设备来查询设备属性。

．GENERIC\_READ：指定对象的读访问，可以读文件的数据且可移动文件中的指针。

．GENERIC\_WRITE：指定对象的写访问，可以写文件的数据且可以移动文件指针，写访问GENERIC\_WRITE要与GENERIC\_READ联合使用。

．dwShoreMode：设成NULL即可。

．lpSecurityAttributes：设成NULL即可。

．dwCreationDisposition：指定对存在的文件采取哪种措施，且当文件不存在时采用哪种措施，此函数必须是下列值中的一个：

．CREAT\_NEW：创建一个新文件，如果文件存在，则函数调用失败。

．CREAT\_ALWAYS：创建一个新文件，如果文件存在，函数重写文件且清空现有属性。

．OPEN\_EXISTING：打开文件，如果文件不存在，则函数调用失败。

．OPEN\_ALWAYS：如果文件存在，则打开文件。如果文件不存在，则创建一个新文件。

．TRUNCATE\_EXISTING：打开文件，一旦文件打开，就被删截掉，从而使文件的大小为0字节，调用函数必须用GENERIC\_WRITE访问来打开文件，如果文件不存在，则函数调用失败。

．dwFlagsAndAttributes：指定文件属性和标志，该参数可取很多种组合，以下示三种：

．FILE\_FLAG\_OVERLAPPED：指导系统对对象进行初始化，以便操作有足够的时间来处理返回ERROR\_IO\_PENDING，当完成操作时，指定事件被设置为发信号状态。

．FILE\_FLAG\_NO\_BUFFERING：引导系统打开没有瞬间缓冲或缓存的文件，当与FILE\_FLAG\_0VERLAPPED结合时，标志给出最大的按时间顺序的操作，因为I／O不依靠内存管理器的时间顺序的操作，但是，因为数据没有在缓存中，一些I/O操作将长一些。

．FILE\_FLAG\_SEQUENTIAL\_SCAN：表明文件从开头到结尾按顺序被访问。使用它，系统可 优化文件缓存。访问方式读大文件的应用程序，指定此标志可以增加它的性能。

．hTemplateFile：设成NULL即可。

返回值：如果函数调用成功，返回值为指向指定文件的打开句柄；如果函数调用失败，返回值为INVALID\_HANDLE\_VALUE。

实验目的

本实验着重于了解磁盘的物理组织，以及如何通过用户态的程序直接调用磁盘I/O API函数（DeviceIoControl）根据输入的驱动器号读取驱动器中磁盘的基本信息，在Windows Server 2003环境进行。

实验内容与参考源代码：

SoftDiskIo-1.cpp

#include <windows.h>

#include <iostream.h>

#include <winioctl.h>// winioctl.h 函数deviceincontrol 用到的许多控制码，都是定义在这里的。

#include <string.h>

struct Disk //关于Disk结构的定义

{

HANDLE handle;

DISK\_GEOMETRY disk\_info;

};

Disk disk;

HANDLE Floppy;

static \_int64 sector;

bool flag;

Disk physicDisk(char driverLetter);

void main(void)

{

char DriverLetter；

cout<< "请输入磁盘号：a/c" <<endl；

cin>>DriverLetter； //选择要查看的磁盘

disk = physicDisk(DriverLetter)；

}

Disk physicDisk(char driverLetter) //

{

flag = true；

DISK\_GEOMETRY\* temp = new DISK\_GEOMETRY；

char device[9] = "\\\\.\\c:"；

device[4] = driverLetter；

Floppy = CreateFile( device, //将要打开的驱动器名

GENERIC\_READ, //存取的权限

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, // 共享的权限

NULL, //默认属性位

OPEN\_EXISTING, //创建驱动器的方式

0, //所创建的驱动器的属性

NULL )； //指向模板文件的句柄

if ( GetLastError() == ERROR\_ALREADY\_EXISTS ) //如打开失败，返回错误代码

{

cout<<"不能打开磁盘"<<endl；

cout<<GetLastError()<<endl；

flag = false；

return disk；

}

DWORD bytereturned；

BOOL Result；

disk.handle = Floppy；

Result = DeviceIoControl ( Floppy,

IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY,

NULL,

0,

temp,

sizeof(\*temp),

&bytereturned,

(LPOVERLAPPED)NULL )；

if (!Result) //如果失败，返回错误代码

{

cout<<"打开失败”<<endl；

cout<<"错误代码为："<<GetLastError()<<endl；

flag = false；

return disk；

}

disk.disk\_info = \*temp； //输出整个物理磁盘的信息

cout<<driverLetter<<"盘有: "<<endl；

cout<<"柱面数为："<<(unsigned long)disk.disk\_info.Cylinders.QuadPart<<endl；

cout<< "每柱面的磁道数为："<<disk.disk\_info.TracksPerCylinder<<endl；

cout<< "每磁道的扇区数为："<<disk.disk\_info.SectorsPerTrack<<endl；

cout<< "每扇区的字节数为："<<disk.disk\_info.BytesPerSector<<endl；

sector = disk.disk\_info.Cylinders.QuadPart\* (disk.disk\_info.TracksPerCylinder)\* (disk.disk\_info.SectorsPerTrack)；

double DiskSize =(double)disk.disk\_info.Cylinders.QuadPart \* //

(disk.disk\_info.TracksPerCylinder) \*

(disk.disk\_info.SectorsPerTrack) \*

(disk.disk\_info.BytesPerSector)；

cout<<driverLetter<<"盘所在磁盘总共有"<<(long)sector<<"个扇区"<<endl；

cout<<"磁盘大为:"<<DiskSize/(1024\*1024)<<"MB "<<endl；

delete temp；

return disk；

}

程序的结果

请输入磁盘号：a/c

a

a盘有:

柱面数为：80

每柱面的磁道数为：2

每磁道的扇区数为：18

每扇区的字节数为：512

a盘所在磁盘总共有2880个扇区

磁盘大为:1.40625MB

讨论

如输入磁盘号为c，显示的磁盘信息是整个硬盘信息，而不是c盘分区的信息。如输入磁盘号为d，显示的磁盘信息与如输入磁盘号为c显示的磁盘信息相同。用磁盘I/O API函数读出的磁盘信息是从硬盘的主引导区得到。