# 这一节我们来学习如何遍历磁盘卷

# 学习大纲

|  |
| --- |
|  |

# 所谓的卷，就是指磁盘分区，一个卷就是一个磁盘分区

# API说明

## 1. GetLogicalDriveStrings

|  |
| --- |
|  |

## 2.GetDriveTpye函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 确定磁盘驱动器是可移动的、固定的、CD-ROM、RAM 磁盘，还是网络驱动器。  若要确定驱动器是否为 USB 类型的驱动器，请调用 [SetupDiGetDeviceRegistryProperty](https://learn.microsoft.com/zh-cn/windows/desktop/api/setupapi/nf-setupapi-setupdigetdeviceregistrypropertya) 并指定 **SPDRP\_REMOVAL\_POLICY** 属性。 语法 C++  UINT GetDriveTypeA(  [in, optional] LPCSTR lpRootPathName  ); 参数 [in, optional] lpRootPathName  驱动器的根目录。  尾随反斜杠是必需的。 如果此参数为 **NULL**，则该函数使用当前目录的根目录。 返回值 返回值指定驱动器的类型，可以是以下值之一。  展开表   |  |  | | --- | --- | | **返回代码/值** | **说明** | | **DRIVE\_UNKNOWN**  0 | 无法确定驱动器类型。 | | **DRIVE\_NO\_ROOT\_DIR**  1 | 根路径无效;例如，没有在指定路径装载的卷。 | | **DRIVE\_REMOVABLE**  2 | 驱动器具有可移动媒体;例如，软盘驱动器、拇指驱动器或闪存卡读卡器。 | | **DRIVE\_FIXED**  3 | 驱动器具有固定介质;例如，硬盘驱动器或闪存驱动器。 | | **DRIVE\_REMOTE**  4 | 该驱动器是远程 (网络) 驱动器。 | | **DRIVE\_CDROM**  5 | 该驱动器是 CD-ROM 驱动器。 | | **DRIVE\_RAMDISK**  6 | 驱动器是一个 RAM 磁盘。 |  备注 在 Windows 8 和 Windows Server 2012 中，此函数由以下技术支持。  展开表   |  |  | | --- | --- | | **技术** | **支持** | | 服务器消息块 (SMB) 3.0 协议 | 否 | | SMB 3.0 透明故障转移 (TFO) | 否 | | 具有横向扩展文件共享的 SMB 3.0 (SO) | 否 | | 群集共享卷文件系统 (CSV) | 是 | | 弹性文件系统 (ReFS) | 是 |     SMB 不支持卷管理功能。  **备注**  fileapi.h 标头将 GetDriveType 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | fileapi.h (包括 Windows.h) | | **Library** | Kernel32.lib | | **DLL** | Kernel32.dll |  另请参阅 [GetDiskFreeSpace](https://learn.microsoft.com/zh-cn/windows/desktop/api/fileapi/nf-fileapi-getdiskfreespacea)  [卷管理函数](https://learn.microsoft.com/zh-cn/windows/desktop/FileIO/volume-management-functions) |

## 3. GetVolumeInformation函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 检索与指定根目录关联的文件系统和卷的相关信息。  若要在检索此信息时指定句柄，请使用 [GetVolumeInformationByHandleW](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getvolumeinformationbyhandlew) 函数。  若要检索文件或目录的当前压缩状态，请使用 [FSCTL\_GET\_COMPRESSION](https://learn.microsoft.com/zh-cn/windows/win32/api/winioctl/ni-winioctl-fsctl_get_compression)。 语法 C++复制  BOOL GetVolumeInformationA(  [in, optional] LPCSTR lpRootPathName,  [out, optional] LPSTR lpVolumeNameBuffer,  [in] DWORD nVolumeNameSize,  [out, optional] LPDWORD lpVolumeSerialNumber,  [out, optional] LPDWORD lpMaximumComponentLength,  [out, optional] LPDWORD lpFileSystemFlags,  [out, optional] LPSTR lpFileSystemNameBuffer,  [in] DWORD nFileSystemNameSize  ); 参数 [in, optional] lpRootPathName  指向字符串的指针，该字符串包含要描述的卷的根目录。  如果此参数为 **NULL**，则使用当前目录的根目录。 需要尾随反斜杠。 例如，将 \\MyServer\MyShare 指定为“\\MyServer\MyShare\”，或将 C 驱动器指定为“C：\”。  [out, optional] lpVolumeNameBuffer  指向接收指定卷名称的缓冲区的指针。 缓冲区大小由 nVolumeNameSize 参数指定。  [in] nVolumeNameSize  卷名称缓冲区的长度（以 **TCHAR 为单位**）。 最大缓冲区大小为 **MAX\_PATH**s+1。  如果未提供卷名称缓冲区，则忽略此参数。  [out, optional] lpVolumeSerialNumber  指向接收卷序列号的变量的指针。  如果不需要序列号，此参数可以为 **NULL** 。  此函数返回格式化硬盘时操作系统分配的卷序列号。 若要以编程方式获取制造商分配的硬盘序列号，请使用 Windows Management Instrumentation (WMI) [Win32\_PhysicalMedia](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/cimwin32a/win32-physicalmedia) 属性 **SerialNumber**。  [out, optional] lpMaximumComponentLength  指向变量的指针，该变量接收指定文件系统支持的文件名组件的最大长度（以 **TCHAR 为单位**）。  文件名组件是文件名在反斜杠之间的部分。  存储在 \*lpMaximumComponentLength 指向的变量中的值用于指示指定的文件系统支持长名称。 例如，对于支持长名称的 FAT 文件系统，函数存储值 255，而不是以前的 8.3 指示器。 使用 NTFS 文件系统的系统上也支持长名称。  [out, optional] lpFileSystemFlags  指向接收与指定文件系统关联的标志的变量的指针。  此参数可以是以下一个或多个标志。 但是， **FILE\_FILE\_COMPRESSION** 和 **FILE\_VOL\_IS\_COMPRESSED** 是相互排斥的。  展开表   | **值** | **含义** | | --- | --- | | **FILE\_CASE\_SENSITIVE\_SEARCH** 0x00000001 | 指定的卷支持区分大小写的文件名。 | | **FILE\_CASE\_PRESERVED\_NAMES** 0x00000002 | 指定卷在磁盘上放置名称时支持保留文件名大小写。 | | **FILE\_UNICODE\_ON\_DISK** 0x00000004 | 指定的卷支持在磁盘上显示的文件名中的 Unicode。 | | **FILE\_PERSISTENT\_ACLS** 0x00000008 | 指定的卷保留并强制实施 ACL) (访问控制列表。 例如，NTFS 文件系统保留并强制实施 ACL，而 FAT 文件系统则不保留。 | | **FILE\_FILE\_COMPRESSION** 0x00000010 | 指定的卷支持基于文件的压缩。 | | **FILE\_VOLUME\_QUOTAS** 0x00000020 | 指定的卷支持磁盘配额。 | | **FILE\_SUPPORTS\_SPARSE\_FILES** 0x00000040 | 指定的卷支持稀疏文件。 | | **FILE\_SUPPORTS\_REPARSE\_POINTS** 0x00000080 | 指定的卷支持重新分析点。  **裁判：** ReFS 支持重新分析点，但不为其编制索引，因此 [FindFirstVolumeMountPoint](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbase-findfirstvolumemountpointa) 和 [FindNextVolumeMountPoint](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbase-findnextvolumemountpointa) 将无法按预期运行。 | | **FILE\_SUPPORTS\_REMOTE\_STORAGE** 0x00000100 | 文件系统支持远程存储。 | | **FILE\_RETURNS\_CLEANUP\_RESULT\_INFO** 0x00000200 | 在成功的清理操作后，文件系统将返回描述清理期间执行的其他操作的信息，例如删除文件。 文件系统筛选器可以在其清理后回调中检查此信息。 | | **FILE\_SUPPORTS\_POSIX\_UNLINK\_RENAME** 0x00000400 | 文件系统支持 POSIX 样式的删除和重命名操作。 | | **FILE\_VOLUME\_IS\_COMPRESSED** 0x00008000 | 指定的卷是压缩卷，例如 DoubleSpace 卷。 | | **FILE\_SUPPORTS\_OBJECT\_IDS** 0x00010000 | 指定的卷支持对象标识符。 | | **FILE\_SUPPORTS\_ENCRYPTION** 0x00020000 | 指定的卷支持加密文件系统 (EFS) 。 有关详细信息，请参阅 [文件加密](https://learn.microsoft.com/zh-cn/windows/win32/FileIO/file-encryption)。 | | **FILE\_NAMED\_STREAMS** 0x00040000 | 指定的卷支持命名流。 | | **FILE\_READ\_ONLY\_VOLUME** 0x00080000 | 指定的卷是只读的。 | | **FILE\_SEQUENTIAL\_WRITE\_ONCE** 0x00100000 | 指定的卷支持单个顺序写入。 | | **FILE\_SUPPORTS\_TRANSACTIONS** 0x00200000 | 指定的卷支持事务。 有关详细信息，请参阅 [关于 KTM](https://learn.microsoft.com/zh-cn/windows/win32/Ktm/about-ktm)。 | | **FILE\_SUPPORTS\_HARD\_LINKS** 0x00400000 | 指定的卷支持硬链接。 有关详细信息，请参阅 [硬链接和交汇点](https://learn.microsoft.com/zh-cn/windows/win32/FileIO/hard-links-and-junctions)。  **Windows Server 2008、Windows Vista、Windows Server 2003 和 Windows XP：** 在 Windows Server 2008 R2 和 Windows 7 之前，不支持此值。 | | **FILE\_SUPPORTS\_EXTENDED\_ATTRIBUTES** 0x00800000 | 指定的卷支持扩展属性。 扩展属性是应用程序特定的元数据片段，应用程序可与文件关联，并且不属于文件数据。  **Windows Server 2008、Windows Vista、Windows Server 2003 和 Windows XP：** 在 Windows Server 2008 R2 和 Windows 7 之前，不支持此值。 | | **FILE\_SUPPORTS\_OPEN\_BY\_FILE\_ID** 0x01000000 | 文件系统支持通过 FileID 打开。 有关详细信息，请参阅 [FILE\_ID\_BOTH\_DIR\_INFO](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/ns-winbase-file_id_both_dir_info)。  **Windows Server 2008、Windows Vista、Windows Server 2003 和 Windows XP：** 在 Windows Server 2008 R2 和 Windows 7 之前，不支持此值。 | | **FILE\_SUPPORTS\_USN\_JOURNAL** 0x02000000 | 指定的卷支持更新序列号 (USN) 日志。 有关详细信息，请参阅[更改日记记录](https://learn.microsoft.com/zh-cn/windows/win32/FileIO/change-journal-records)。  **Windows Server 2008、Windows Vista、Windows Server 2003 和 Windows XP：** 在 Windows Server 2008 R2 和 Windows 7 之前，不支持此值。 | | **FILE\_SUPPORTS\_INTEGRITY\_STREAMS** 0x04000000 | 文件系统支持 [完整性流](https://learn.microsoft.com/zh-cn/windows-server/storage/refs/integrity-streams)。 | | **FILE\_SUPPORTS\_BLOCK\_REFCOUNTING** 0x08000000 | 指定的卷支持在同一卷上的文件之间共享逻辑群集。 文件系统在写入共享群集时重新分配。 指示 **FSCTL\_DUPLICATE\_EXTENTS\_TO\_FILE** 是受支持的操作。 | | **FILE\_SUPPORTS\_SPARSE\_VDL** 0x10000000 | 文件系统跟踪文件的每个群集是否包含来自显式文件写入或自动零的有效数据 () 或无效数据 (尚未写入或归零) 。 使用稀疏有效数据长度 (VDL) 的文件系统不存储有效数据长度，也不需要有效数据在文件中连续。 | | **FILE\_DAX\_VOLUME** 0x20000000 | 指定的卷是 DAX) 卷 (直接访问。  **注意：** 此标志是在 Windows 10 版本 1607 中引入的。 | | **FILE\_SUPPORTS\_GHOSTING** 0x40000000 | 文件系统支持重影。 |   [out, optional] lpFileSystemNameBuffer  指向接收文件系统名称的缓冲区的指针，例如 FAT 文件系统或 NTFS 文件系统。 缓冲区大小由 nFileSystemNameSize 参数指定。  [in] nFileSystemNameSize  文件系统名称缓冲区的长度（以 **TCHAR 为单位**）。 最大缓冲区大小为 **MAX\_PATH**+1。  如果未提供文件系统名称缓冲区，则忽略此参数。 返回值 如果检索了所有请求的信息，则返回值为非零值。  如果未检索所有请求的信息，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。 注解 当用户尝试获取有关没有软盘的软盘驱动器或没有光盘的 CD-ROM 驱动器的信息时，系统会显示一个消息框，供用户分别插入软盘或光盘。 若要防止系统显示此消息框，请使用 **SEM\_FAILCRITICALERRORS**调用 [SetErrorMode](https://learn.microsoft.com/zh-cn/windows/win32/api/errhandlingapi/nf-errhandlingapi-seterrormode) 函数。  **FILE\_VOL\_IS\_COMPRESSED**标志是基于卷的压缩的唯一指示器。 文件系统名称不会更改以指示压缩，例如，此标志在 DoubleSpace 卷上返回设置。 如果压缩是基于卷的，则会压缩或未压缩整个卷。  **FILE\_FILE\_COMPRESSION**标志指示文件系统是否支持基于文件的压缩。 当压缩基于文件时，可以压缩或不压缩单个文件。  **FILE\_FILE\_COMPRESSION**和**FILE\_VOL\_IS\_COMPRESSED**标志互斥。 不能设置这两个位。  存储在 lpMaximumComponentLength 中的最大组件长度值是唯一指示卷支持长于正常 FAT 文件系统 (或其他文件系统) 文件名。 不会更改文件系统名称以指示对长文件名的支持。  [GetCompressedFileSize](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getcompressedfilesizea) 函数获取文件的压缩大小。 [GetFileAttributes](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getfileattributesa) 函数可以确定是否压缩单个文件。 符号链接行为 如果路径指向符号链接，则函数会返回目标的卷信息。  从Windows 8和Windows Server 2012开始，以下技术支持此函数。  展开表   | **技术** | **支持** | | --- | --- | | 服务器消息块 (SMB) 3.0 协议 | 否 | | SMB 3.0 透明故障转移 (TFO) | 否 | | 具有横向扩展文件共享的 SMB 3.0 (SO) | 否 | | 群集共享卷文件系统 (CSV) | 是 | | 弹性文件系统 (ReFS) | 是 |   SMB 不支持卷管理功能。 事务处理操作 如果卷支持文件系统事务，则函数返回 lpFileSystemFlags 中的 FILE\_SUPPORTS\_TRANSACTIONS。  **备注**  fileapi.h 标头将 GetVolumeInformation 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名的使用与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | fileapi.h (包括 Windows.h) | | **Library** | Kernel32.lib | | **DLL** | Kernel32.dll |  另请参阅 [关于 KTM](https://learn.microsoft.com/zh-cn/windows/win32/Ktm/about-ktm)  [文件加密](https://learn.microsoft.com/zh-cn/windows/win32/FileIO/file-encryption)  [GetCompressedFileSize](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getcompressedfilesizea)  [GetFileAttributes](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getfileattributesa)  [GetVolumeInformationByHandleW](https://learn.microsoft.com/zh-cn/windows/win32/api/fileapi/nf-fileapi-getvolumeinformationbyhandlew)  [SetErrorMode](https://learn.microsoft.com/zh-cn/windows/win32/api/errhandlingapi/nf-errhandlingapi-seterrormode)  [SetVolumeLabel](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbase-setvolumelabela)  [卷管理函数](https://learn.microsoft.com/zh-cn/windows/win32/FileIO/volume-management-functions) |

## 演练，

## 1.创建一个c++常规空项目，取名：lesson12-travel-disk-volume，然后新建一个cpp源文件，先学习GetLogicalDriveStrings，有2个代码实例

## 2.实例1

### travel-volume.cpp

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #include<stdlib.h>  #define BUF\_SIZE 1024  //注意：这里有坑，使用vs2008编程，最好先把所有变量在上面声明，然后再在下面赋值，否则结果不对  int main()  {  CHAR driveStrs[BUF\_SIZE];  PCHAR pDrive;  ZeroMemory(driveStrs,BUF\_SIZE);  DWORD dwSize = GetLogicalDriveStrings(BUF\_SIZE-1,driveStrs);  if(!dwSize)  {  MessageBox(NULL,"get drive strings failed","error",0);  }    //printf("drive strings: %s\n",driveStrs);  pDrive = (PCHAR)driveStrs;  printf("logical drives:\n");  do  {  printf("%s\n",pDrive);  pDrive +=(lstrlen(pDrive)+1);  }while(\*pDrive!='\x00');  system("pause");  return 0;  } |

### 效果：

|  |
| --- |
|  |

### 这里有宽字符版本

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #define BUF\_SIZE 1024  int main()  {  WCHAR driveStrs[BUF\_SIZE];  ZeroMemory(driveStrs, BUF\_SIZE);  WCHAR\* pDrive;  GetLogicalDriveStrings(BUF\_SIZE - 1, driveStrs);  pDrive = (WCHAR\*)driveStrs;  wprintf\_s(L"Logical Drives:\n");  do  {  wprintf\_s(L"%s:\n",pDrive);  pDrive += lstrlen(pDrive) + 1;  } while (\*pDrive != L'\x00');  system("pause");  return 0;  } |

## 3.实例2.

### travel-volume.cpp

|  |
| --- |
| #include <Windows.h>  #include <stdio.h>  #include <stdlib.h>  #define BUFSIZE 1024  BOOL GetDriverInfo(LPSTR szDrive);  //int WinMain(HINSTANCE hInstance,HINSTANCE hPreInstance,LPSTR lpCmdLine,int nCmdShow)  int main(void)  {  CHAR szLogicDriveStrings[BUFSIZE];  PCHAR szDrive;  ZeroMemory(szLogicDriveStrings,BUFSIZE);  GetLogicalDriveStrings(BUFSIZE-1,szLogicDriveStrings);  szDrive = (PCHAR)szLogicDriveStrings;  do  {  if(!GetDriverInfo(szDrive))  {  printf("\nGet Volume Information Error:%d",GetLastError());  }  szDrive += (lstrlen(szDrive)+1);  } while (\*szDrive !='\x00');  system("PAUSE");  return 0;  }  BOOL GetDriverInfo(LPSTR szDrive)  {  UINT uDriverType;  DWORD dwVolumeSerialNumber;  DWORD dwMaximumComponentlength;  DWORD dwFileSystemFlags;  CHAR szFileSystemNameBuffer[BUFSIZE];  CHAR szDriveName[MAX\_PATH];  printf("\n%s\n",szDrive);  uDriverType = GetDriveType(szDrive);  switch(uDriverType)  {  case DRIVE\_UNKNOWN:  printf("The driver type cannot be determined!");  break;  case DRIVE\_NO\_ROOT\_DIR:  printf("The root path is invalid,for example,no volume is mounted at the path");  break;  case DRIVE\_REMOVABLE:  printf("The drive is a type that has removable media,for example:a floppy drive or removable hard disk");  break;  case DRIVE\_FIXED:  printf("The drive is a type that cannot be removed, for example,a fixed hard drive");  break;  case DRIVE\_REMOTE:  printf("This drive is a remote(network) drive");  break;  case DRIVE\_CDROM:  printf("This drive is a CD-ROM drive.");  break;  case DRIVE\_RAMDISK:  printf("This drive is a RAM disk");  break;  default:  break;  }  if (!(GetVolumeInformation(  szDrive,  szDriveName,  MAX\_PATH,  &dwVolumeSerialNumber,  &dwMaximumComponentlength,  &dwFileSystemFlags,  szFileSystemNameBuffer,  BUFSIZE)))  {  return FALSE;  }  if (0!=lstrlen(szDriveName))  {  printf("\nDrive Name is %s.\n",szDriveName);  }  printf("\nVolume Serial is %u.",dwVolumeSerialNumber );  printf("\nMaximum Component Length is %u.",dwMaximumComponentlength);  printf("\nSystem Type is %s.\n",szFileSystemNameBuffer);    if (dwFileSystemFlags & FILE\_VOLUME\_QUOTAS)  {  printf("The file system supports disk Quotas.\n");  }  if (dwFileSystemFlags & FILE\_SUPPORTS\_REPARSE\_POINTS)  {  printf("The file system does not support volume mount points.\n");  }  if (dwFileSystemFlags & FILE\_CASE\_SENSITIVE\_SEARCH)  {  printf("The file system supports case-sentitive file name.\n");  }  printf("...\n");  return TRUE;  } |

## 效果：

|  |
| --- |
| C:\  The drive is a type that cannot be removed, for example,a fixed hard drive  Volume Serial is 3363692264.  Maximum Component Length is 255.  System Type is NTFS.  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ...  D:\  The drive is a type that cannot be removed, for example,a fixed hard drive  Drive Name is 数据盘.  Volume Serial is 1550604159.  Maximum Component Length is 255.  System Type is NTFS.  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ...  E:\  The drive is a type that cannot be removed, for example,a fixed hard drive  Drive Name is 新加卷.  Volume Serial is 542590885.  Maximum Component Length is 255.  System Type is NTFS.  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ...  F:\  The drive is a type that cannot be removed, for example,a fixed hard drive  Drive Name is Document and software.  Volume Serial is 3639735162.  Maximum Component Length is 255.  System Type is NTFS.  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ...  G:\  The drive is a type that cannot be removed, for example,a fixed hard drive  Drive Name is learning.  Volume Serial is 3629183472.  Maximum Component Length is 255.  System Type is NTFS.  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name. |

## 这里有宽字符版本

|  |
| --- |
| #include <Windows.h>  #include <stdio.h>  #include <stdlib.h>  #include<locale.h>  #define BUFSIZE 1024  BOOL OutputDriveInfo(LPWSTR szDrive);  int main()  {  setlocale(LC\_ALL, "");  WCHAR driveStrs[BUFSIZE];  WCHAR\* pDrive;  ZeroMemory(driveStrs, BUFSIZE);  GetLogicalDriveStrings(BUFSIZE - 1, driveStrs);  pDrive = (WCHAR\*)driveStrs;  do  {  if (!OutputDriveInfo(pDrive))  {  wprintf\_s(L"Get Volume information Failed,error code:%d\n",GetLastError());  }  pDrive += lstrlen(pDrive) + 1;  } while (\*pDrive != L'\x00');  system("pause");  return 0;  }  BOOL OutputDriveInfo(LPWSTR szDrive)  {  UINT driveType;  DWORD dwVolumeSerialNumber;  DWORD dwMaximumComponentlength;  DWORD dwFileSystemFlags;  WCHAR szFileSystemNameBuffer[BUFSIZE];  WCHAR szDriveName[MAX\_PATH];  wprintf\_s(L"%s\n",szDrive);  driveType = GetDriveType(szDrive);//获取逻辑驱动器类型  switch (driveType)  {  case DRIVE\_UNKNOWN:  wprintf\_s(L"Drive type:Unknow Drive Type\n");  break;  case DRIVE\_REMOVABLE:  wprintf\_s(L"Drive type:Removable Drive \n");  break;  case DRIVE\_FIXED:  wprintf\_s(L"Drive type:HARD Drive \n");  break;  case DRIVE\_NO\_ROOT\_DIR:  wprintf\_s(L"Drive type:Invalid Drive \n");  break;  case DRIVE\_REMOTE:  wprintf\_s(L"Drive type:Network Drive \n");  break;  case DRIVE\_CDROM:  wprintf\_s(L"Drive type:This drive is a CD-ROM drive.");  break;  case DRIVE\_RAMDISK:  wprintf\_s(L"Drive type:This drive is a RAM disk");  break;  default:  break;  }  if (!GetVolumeInformation(szDrive, szDriveName, BUFSIZE, &dwVolumeSerialNumber, &dwMaximumComponentlength,  &dwFileSystemFlags, szFileSystemNameBuffer, BUFSIZE))  {  wprintf\_s(L"Get the volume information Failed,error code:%d",GetLastError());  return FALSE;  }  wprintf\_s(L"drive name:%s\n", szDriveName);  wprintf\_s(L"Volume Serial Number:%d\n", dwVolumeSerialNumber);  wprintf\_s(L"Maximum Component Length:%d\n", dwMaximumComponentlength);  wprintf\_s(L"File System Flags:%d\n", dwFileSystemFlags);  wprintf\_s(L"File System Name:%s\n", szFileSystemNameBuffer);  if (dwFileSystemFlags & FILE\_VOLUME\_QUOTAS)  {  wprintf\_s(L"The file system supports disk Quotas.\n");  }  if (dwFileSystemFlags & FILE\_SUPPORTS\_REPARSE\_POINTS)  {  wprintf\_s(L"The file system does not support volume mount points.\n");  }  if (dwFileSystemFlags & FILE\_CASE\_SENSITIVE\_SEARCH)  {  wprintf\_s(L"The file system supports case-sentitive file name.\n");  }  return TRUE;  } |

## 演练2.FindFirstVolume和FindNextVolume还有FindVolumeClose

## 1.新建一个常规空项目，取名Lesson12\_2-findVolume，添加一个cpp文件，添加一些代码

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #include<stdlib.h>  int main()  {  HANDLE hVol;  CHAR szVolume[MAX\_PATH] = { 0 };  hVol = FindFirstVolumeA(szVolume,MAX\_PATH);  if(hVol == INVALID\_HANDLE\_VALUE)  {  MessageBox(NULL,"Find First Volume Failed...","error",0);  }  printf("First Volume:%s\n",szVolume);  FindVolumeClose(hVol);  system("pause");  return 0;  } |

### 编译运行程序，效果如下

|  |
| --- |
|  |

### 得到的结果和我预期的不一样。那这么处理这种数据呢？这是一个设备名称，这是window的GUID标记。

## 2.我们可以利用循环调用FindNextVolume来找到下一个，代码如下

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #include<stdlib.h>  int main()  {  HANDLE hVol;  CHAR szVolume[MAX\_PATH] = { 0 };  hVol = FindFirstVolumeA(szVolume,MAX\_PATH);  BOOL bFlag;  if(hVol == INVALID\_HANDLE\_VALUE)  {  MessageBox(NULL,"Find First Volume Failed...","error",0);  }  printf("First Volume:%s\n",szVolume);  while(FindNextVolume(hVol,szVolume,MAX\_PATH))  {  printf("NextVolume:%s\n",szVolume);  }  bFlag = FindVolumeClose(hVol);  system("pause");  return bFlag;  } |

### 效果：

|  |
| --- |
|  |

### 这里有一个宽字符版本的程序

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #include<stdlib.h>  int main()  {  BOOL bFlag;  HANDLE hVol;  WCHAR wcVolumeName[MAX\_PATH];  ZeroMemory(wcVolumeName, MAX\_PATH);  hVol = FindFirstVolume(wcVolumeName, MAX\_PATH);  if (hVol == INVALID\_HANDLE\_VALUE)  {  wprintf\_s(L"Find First Volume Failed,error:%d\n", GetLastError());  return -1;  }  do {  wprintf\_s(L"%s\n", wcVolumeName);  } while (FindNextVolume(hVol, wcVolumeName, MAX\_PATH));  //下面是另外一种写法,用while代替do-while也是对的  /\*wprintf\_s(L"First Volume:%s\n", wcVolumeName);  while (FindNextVolume(hVol, wcVolumeName, MAX\_PATH))  {  wprintf\_s(L"Next Volume:%s\n", wcVolumeName);  }\*/  bFlag = FindVolumeClose(hVol);  system("pause");  return bFlag;  } |

### 那么问题来了，这些数据这么使用？

### 其实他们和”c:\\”是一样的用法,作为参数直接传递给GetVolumeInformation函数,也能够获取每一个分卷信息.看看下面的程序

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  #include<stdlib.h>  #include<locale.h>  #define BUFSIZE 1024  BOOL OutputVolumeInfo(LPWSTR szDrive)  {  UINT driveType;  DWORD dwVolumeSerialNumber;  DWORD dwMaximumComponentlength;  DWORD dwFileSystemFlags;  WCHAR szFileSystemNameBuffer[BUFSIZE];  WCHAR szDriveName[MAX\_PATH];  wprintf\_s(L"%s\n", szDrive);  driveType = GetDriveType(szDrive);//获取逻辑驱动器类型  switch (driveType)  {  case DRIVE\_UNKNOWN:  wprintf\_s(L"Drive type:Unknow Drive Type\n");  break;  case DRIVE\_REMOVABLE:  wprintf\_s(L"Drive type:Removable Drive \n");  break;  case DRIVE\_FIXED:  wprintf\_s(L"Drive type:HARD Drive \n");  break;  case DRIVE\_NO\_ROOT\_DIR:  wprintf\_s(L"Drive type:Invalid Drive \n");  break;  case DRIVE\_REMOTE:  wprintf\_s(L"Drive type:Network Drive \n");  break;  case DRIVE\_CDROM:  wprintf\_s(L"Drive type:This drive is a CD-ROM drive.");  break;  case DRIVE\_RAMDISK:  wprintf\_s(L"Drive type:This drive is a RAM disk");  break;  default:  break;  }  if (!GetVolumeInformation(szDrive, szDriveName, BUFSIZE, &dwVolumeSerialNumber, &dwMaximumComponentlength,  &dwFileSystemFlags, szFileSystemNameBuffer, BUFSIZE))  {  wprintf\_s(L"Get the volume information Failed,error code:%d", GetLastError());  return FALSE;  }  wprintf\_s(L"drive name:%s\n", szDriveName);  wprintf\_s(L"Volume Serial Number:%d\n", dwVolumeSerialNumber);  wprintf\_s(L"Maximum Component Length:%d\n", dwMaximumComponentlength);  wprintf\_s(L"File System Flags:%d\n", dwFileSystemFlags);  wprintf\_s(L"File System Name:%s\n", szFileSystemNameBuffer);  if (dwFileSystemFlags & FILE\_VOLUME\_QUOTAS)  {  wprintf\_s(L"The file system supports disk Quotas.\n");  }  if (dwFileSystemFlags & FILE\_SUPPORTS\_REPARSE\_POINTS)  {  wprintf\_s(L"The file system does not support volume mount points.\n");  }  if (dwFileSystemFlags & FILE\_CASE\_SENSITIVE\_SEARCH)  {  wprintf\_s(L"The file system supports case-sentitive file name.\n");  }  wprintf\_s(L"===================================\n");  return TRUE;  }  int main()  {  setlocale(LC\_ALL, "");  BOOL bFlag;  HANDLE hVol;  WCHAR wcVolumeName[MAX\_PATH];  ZeroMemory(wcVolumeName, MAX\_PATH);  hVol = FindFirstVolume(wcVolumeName, MAX\_PATH);  if (hVol == INVALID\_HANDLE\_VALUE)  {  wprintf\_s(L"Find First Volume Failed,error:%d\n", GetLastError());  return -1;  }  do {  // wprintf\_s(L"%s\n", wcVolumeName);  OutputVolumeInfo(wcVolumeName);  } while (FindNextVolume(hVol, wcVolumeName, MAX\_PATH));  //下面是另外一种写法,用while代替do-while也是对的  /\*wprintf\_s(L"First Volume:%s\n", wcVolumeName);  while (FindNextVolume(hVol, wcVolumeName, MAX\_PATH))  {  wprintf\_s(L"Next Volume:%s\n", wcVolumeName);  }\*/  bFlag = FindVolumeClose(hVol);  system("pause");  return bFlag;  } |

### 效果是一样的,只是这一长串不太好看

|  |
| --- |
| \\?\Volume{ef2208ef-a016-453c-89ec-1ad83fbddc4e}\  Drive type:HARD Drive  drive name:  Volume Serial Number:-931275032  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{3749bab7-e4b4-4db3-9706-5e85f2594219}\  Drive type:HARD Drive  drive name:  Volume Serial Number:-59590357  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{c701f658-4cc2-4b1c-849f-30b9943704da}\  Drive type:HARD Drive  drive name:新加卷  Volume Serial Number:542590885  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{e6f96d59-79ac-46cd-a157-b27772aa7798}\  Drive type:HARD Drive  drive name:数据盘  Volume Serial Number:1550604159  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{68d7ac8d-4ac9-4848-9050-ea804525c46a}\  Drive type:HARD Drive  drive name:WINRE\_DRV  Volume Serial Number:2023439685  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{4bc244d6-0000-0000-0000-100000000000}\  Drive type:HARD Drive  drive name:Document and software  Volume Serial Number:-655232134  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{4bc244d6-0000-0000-0000-307ddc000000}\  Drive type:HARD Drive  drive name:learning  Volume Serial Number:-665783824  Maximum Component Length:255  File System Flags:65472255  File System Name:NTFS  The file system supports disk Quotas.  The file system does not support volume mount points.  The file system supports case-sentitive file name.  ===================================  \\?\Volume{e1e06319-4b6c-48c8-814f-437507cc1285}\  Drive type:HARD Drive  drive name:SYSTEM\_DRV  Volume Serial Number:1352108897  Maximum Component Length:255  File System Flags:131590  File System Name:FAT32  ===================================  请按任意键继续. . . |