<https://www.cnblogs.com/findumars/p/6253630.html>

linux下可以使用命令df -h来获取各个(已加载)分区的使用情况。Windows下也有很多好的工具来获取，但是我没有发现windows下的df命令。

在linux下使用df -h命令的输出如下

o@Neo-kylin:~/snmp$ df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda2 197G 14G 174G 8% /

tmpfs 922M 76K 922M 1% /dev/shm

/dev/sda5 61G 7.8G 50G 14% /media/sda5

/dev/sda6 134G 29G 99G 23% /media/sda6

在windows下获取这些信息可以通过几个API函数来操作。

**GetLogicalDriveStrings函数**

Windows的API函数名称一般都很长，虽然不好记，但是描述的意思比较清晰。这个函数就是用于获取逻辑驱动器字符串。

GetLogicalDriveStrings实际上是一个宏，在没有定义UNICODE宏的条件下，它被替换为GetLogicalDriveStringA函数，在定义了UNICODE宏的条件下，它被替换为GetLogicalDriveStringsW函数。

这两个函数的声明如下

DWORD GetLogicalDriveStringsA( DWORD nBufferLength, \_Out\_writes\_to\_opt\_(nBufferLength, return + 1) LPSTR lpBuffer );

DWORD

GetLogicalDriveStringsW(

DWORD nBufferLength,

\_Out\_writes\_to\_opt\_(nBufferLength, return + 1) LPWSTR lpBuffer

);

这个的参数看起来很复杂，其实并没有。函数需要提供一个内存缓冲区lpBuffer来供它保存获取的逻辑驱动器的分区号（C:\ ,D:\等）信息。

如果参数nBufferLength填写0，那么将缓冲区将不使用，函数返回保存所有数据所需要的字节数。这通常用户获取需要的缓冲区大小。

应该总是比较返回值与参数nBufferLength的大小。

如果函数成功，返回值是复制到缓冲区的字符串的长度， 不包括结束符null。注意，ansi-ascii的null字符用一个字节，但 Unicode（UTF-16）null字符用两个字节。

如果缓冲区不够大，返回值是大于nbufferlength。它要求具有能够保持驱动字符串大小的缓冲区。

如果函数失败，返回值是零。为了获得更多的错误信息，可以使用GetLastError函数。

这里不讲UNICODE与多字节字符集的区别。指导一点就好，使用多字节字符集的时候，当作普通的C风格字符串来使用即可。

**使用示例**

**获取需要的缓冲区长度示例**

#include <stdio.h>

#include <Windows.h>

int main()

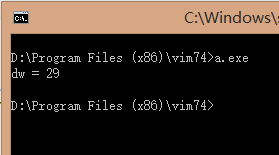
{

DWORD dw = GetLogicalDriveStrings(0,NULL);

printf("dw = %lu\n",dw); return 0;

}

编译后运行输出



**获取所有驱动器号示例**

#include <stdio.h>

#include <Windows.h>

int main()

{

DWORD dwSize = MAX\_PATH;

char szLogicalDrives[MAX\_PATH] = {0};

//获取逻辑驱动器号字符串

DWORD dwResult = GetLogicalDriveStrings(dwSize,szLogicalDrives);

//处理获取到的结果

if (dwResult > 0 && dwResult <= MAX\_PATH) {

char\* szSingleDrive = szLogicalDrives; //从缓冲区起始地址开始

while(\*szSingleDrive) {

printf("Drive: %s\n", szSingleDrive); //输出单个驱动器的驱动器号

// 获取下一个驱动器号起始地址

szSingleDrive += strlen(szSingleDrive) + 1;

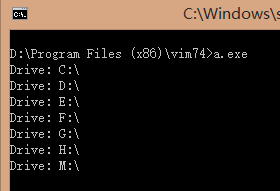
}

}

return 0;

}

编译后运行输出



**GetDriveType函数**

GetDriveType函数用于判断一个磁盘驱动器的类型。  
函数声明如下

UINT WINAPI GetDriveType(

\_In\_opt\_ LPCTSTR lpRootPathName

);

参数lpRootPathName包含了根目录路径的字符串指针。  
如驱动器不能识别，则返回零。如指定的目录不存在，则返回1。如执行成功，则用下述任何一个常数指定驱动器类型

| **常数** | **含义** |
| --- | --- |
| DRIVE\_UNKNOWN | 未知的磁盘类型 |
| DRIVE\_NO\_ROOT\_DIR | 说明lpRootPathName是无效的 |
| DRIVE\_REMOVABLE | 可移动磁盘 |
| DRIVE\_FIXED | 固定磁盘 |
| DRIVE\_REMOTE | 网络磁盘 |
| DRIVE\_CDROM | 光驱 |
| DRIVE\_RAMDISK | RAM映射磁盘 |

**使用示例**

获取所有驱动器号及其所属磁盘类型示例

输出逻辑驱动器类型函数

#include <stdio.h>

#include <Windows.h>

void putDrivesType(const char\* lpRootPathName)

{

UINT uDriverType = GetDriveType(lpRootPathName);

switch(uDriverType) {

case DRIVE\_UNKNOWN :puts("未知的磁盘类型"); break;

case DRIVE\_NO\_ROOT\_DIR: puts("路径无效"); break;

case DRIVE\_REMOVABLE: puts("可移动磁盘"); break;

case DRIVE\_FIXED: puts("固定磁盘"); break;

case DRIVE\_REMOTE: puts("网络磁盘"); break;

case DRIVE\_CDROM: puts("光驱"); break;

case DRIVE\_RAMDISK: puts("内存映射盘"); break;

default:

break;

}

}

调用

int main()

{

DWORD dwSize = MAX\_PATH;

char szLogicalDrives[MAX\_PATH] = {0};

//获取逻辑驱动器号字符串

DWORD dwResult = GetLogicalDriveStrings(dwSize,szLogicalDrives);

//处理获取到的结果

if (dwResult > 0 && dwResult <= MAX\_PATH) {

char\* szSingleDrive = szLogicalDrives; //从缓冲区起始地址开始

while(\*szSingleDrive) {

printf("Drive: %s\n", szSingleDrive); //输出单个驱动器的驱动器号

putDrivesType(szSingleDrive); //输出逻辑驱动器类型

// 获取下一个驱动器号起始地址

szSingleDrive += strlen(szSingleDrive) + 1;

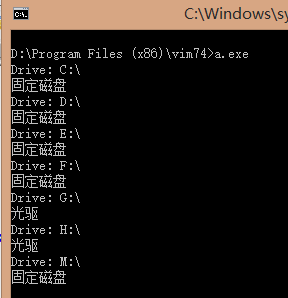
}

}

return 0;

}

编译后运行输出



**GetDiskFreeSpaceEx 函数**

GetDiskFreeSpaceEx函数用户获取逻辑驱动器的容量信息。还有一个和它长得很像的函数GetDiskFreeSpace，但这个函数已经过时了，不推荐使用。

函数声明如下

BOOL WINAPI GetDiskFreeSpaceEx(

\_In\_ LPCTSTR lpRootPathName,

\_Out\_ LPDWORD lpSectorsPerCluster,

\_Out\_ LPDWORD lpBytesPerSector,

\_Out\_ LPDWORD lpNumberOfFreeClusters,

\_Out\_ LPDWORD lpTotalNumberOfClusters

);

这个函数的参数要仔细的说明一下。

| **参数** | **含义** |
| --- | --- |
| lpDirectoryName | 逻辑驱动器的名称(C/D/E等这些) |
| lpFreeBytesAvailableToCaller | 用户(当前线程)可用的磁盘空间字节数 |
| lpTotalNumberOfBytes | 逻辑磁盘总的空间字节数 |
| lpTotalNumberOfFreeBytes | 逻辑磁盘空闲的空间字节数 |

上面三个字节数的单位都是字节，数据类型都是64位无符号整型。

GetDiskFreeSpaceEx函数执行成功返回非0值，失败返回0。可以通过GetLastError函数获取失败信息。

**使用示例**

获取磁盘容量信息示例

下面的函数用来输出磁盘的容量信息。

void putDrivesFreeSpace(const char\* lpRootPathName)

{

unsigned long long available,total,free;

if(GetDiskFreeSpaceEx(lpRootPathName,(ULARGE\_INTEGER\*)&available,(ULARGE\_INTEGER\*)&total,(ULARGE\_INTEGER\*)&free)){

printf("Drives %s | total = %lld MB,available = %lld MB,free = %lld MB\n",

lpRootPathName,total>>20,available>>20,free>>20);

}else{

puts("获取容量信息失败");

}

}

调用如下

int main()

{

DWORD dwSize = MAX\_PATH;

char szLogicalDrives[MAX\_PATH] = {0};

//获取逻辑驱动器号字符串

DWORD dwResult = GetLogicalDriveStrings(dwSize,szLogicalDrives);

//处理获取到的结果

if (dwResult > 0 && dwResult <= MAX\_PATH) {

char\* szSingleDrive = szLogicalDrives; //从缓冲区起始地址开始

while(\*szSingleDrive) {

printf("Drive: %s\n", szSingleDrive); //输出单个驱动器的驱动器号

putDrivesType(szSingleDrive); //输出逻辑驱动器类型

putDrivesFreeSpace(szSingleDrive);

// 获取下一个驱动器号起始地址

szSingleDrive += strlen(szSingleDrive) + 1;

}

}

return 0;

}

编译后运行输出  


http://www.cnblogs.com/oloroso/p/4881394.html