How to match linux device path to windows drive name?

Asked 16 years, 4 months ago Modified 14 years, 10 months ago
Viewed 6k times

3

I'm writing an application that on some stage performs low-level disk operations in Linux environment. The app actually consists of 2 parts, one runs on Windows and interacts with a user and another is a linux part that runs from a LiveCD. User makes a choice of Windows drive letters and then a linux part performs actions with corresponding partitions. The problem is finding a match between a Windows drive letter (like C:) and a linux device name (like /dev/sda1). This is my current solution that I rate as ugly:

- store partitions information (i.e. drive letter, number of blocks, drive serial number etc.) in Windows in some pre-defined place (i.e. the root of the system partition).
- read a list of partitions from /proc/partitions. Get only those partitions that has major number for SCSI or IDE hard drives and minor number that identifies them as real partitions and not the whole disks.
- Try to mount each of them with either ntfs or vfat file systems. Check whether the mounted partition contains the information stored by Windows app.

- Upon finding the required information written by the Windows app make the actual match. For each partition found in /proc/partitions acquire drive serial number (via HDIO_GET_IDENTITY syscall), number of blocks (from /proc/partitions) and drive offset (/sys/blocks/drive_path/partition_name/start), compare this to the Windows information and if this matches - store a Windows drive letter along with a linux device name.

There are a couple of problems in this scheme:

- This is ugly. Writing data in Windows and then reading it in Linux makes testing a nightmare.
- linux device major number is compared only with IDE or SCSI devices. This would probably fail, i.e. on USB or FireWire disks. It's possible to add these types of disks, but limiting the app to only known subset of possible devices seems to be rather bad idea.
- looks like HDIO_GET_IDENTITY works only on IDE and SATA drives.
- /sys/block hack may not work on other than IDE or SATA drives.

Any ideas on how to improve this schema? Perhaps there is another way to determine windows names without writing all the data in windows app?

P.S. The language of the app is C++. I can't change this.

c++ linux drives

Share

Improve this question

Follow

edited Sep 11, 2008 at 14:13

Community Bot

1 ● 1

asked Aug 20, 2008 at 22:06

alex

43 ● 2 ● 4

6 Answers

Sorted by: Highest score (default)

2

Partitions have UUIDs associated with them

My knowledge of this is very shallow, but I thought that was only true for disks formatted with GPT (Guid Partition Table) partitions, rather than the old-style MBR format which 99% of the world is still stuck with?

Share Improve this answer

Follow

answered Aug 20, 2008 at 22:53

Orion Edwards

123k ● 66 ● 245 ● 339

1

Partitions have UUIDs associated with them. I don't know how to find these in Windows but in linux you can find the UUID for each partition with:

> sudo vol_id -u device (e.g. /dev/sda1)

If there is an equivilent function in Windows you could simply store the UUIDs for whatever partition they pick then iterate through all known partitions in linux and match the UUIDs.

Edit: This may be a linux-only thing, and it may speficially be the volid util that generates these from something (instead of reading off meta-data for the drive). Having said that, there is nothing stopping you getting the source for volid and checking out what it does.

Share Improve this answer Follow

edited Aug 20, 2008 at 22:25

answered Aug 20, 2008 at 22:20

SCdF

59.3k ● 24 ● 79 ● 114

1

> My knowledge of this is very shallow, but I thought that was only true for disks formatted with GPT (Guid Partition Table) partitions, rather than the old-style MBR format which 99% of the world is still stuck with?

Not to sounds like a linux user cliche but it Works For Me.. I use it with NTFS partitions and have had no problems. As I said in my edit, vol_id may be generating

them itself. If that were the case there would be no reliance on any particular partition format, which would be swell.

Share Improve this answer Follow

answered Aug 21, 2008 at 0:38

SCdF

59.3k • 24 • 79 • 114

1

> Partitions have UUIDs associated with them. I don't know how to find these in Windows but in linux you can find the UUID for each partition with:
>
> > sudo vol_id -u device (e.g. /dev/sda1)
>
> If there is an equivilent function in Windows you could simply store the UUIDs for whatever partition they pick then iterate through all known partitions in linux and match the UUIDs.

That's a good point, thank you! I've looked to the sources of vol_id (a part of the udev tarball) and it seems that for FAT(32) and NTFS it generates UUUD using the volume serial number that is read from the predefined location on the partition. Since I don't expect anything other then fat32 and ntfs I consider to use this information as a partition identifier.

Share Improve this answer

answered Aug 21, 2008 at 7:37

Follow

alex
43 ● 2 ● 4

0

You need to either mark the drive in some way (e.g. write a file etc.), or find some identifier that is only associated with that particular drive.

It is very hard, almost impossible to figure out what letter Windows would assign to a particular drive partition, without actually running Windows. This is because Windows always associates the drive that it is run from with C:. Which could be any drive, if you have more than one operating system installed. Windows also allows you to choose what drive letter it will try first, for a specific partition, causing further problems.

It would be a whole lot easier to do the GUI stuff inside Linux, than to try this mixed Window/Linux solution. I'm not say don't try it this way, what I am saying is there are very many possible pitfalls with this approach. I'm sure I don't even know about all of them.

Another option would be to see if you could actually do the Linux part, inside of Windows. If you are a very good Windows programmer, you can actually get access to the raw file-system. There are probably just as many pitfalls with this approach, because Windows will be running while all of this is in operation.

So to re-iterate I would see if you could do everything from within Linux, if you can. It's just a whole lot simpler in

the long run.

Share Improve this answer Follow

answered Aug 21, 2008 at 0:49

Brad Gilbert

34.1k ● 11 ● 79 ● 130

0

In Windows you can read the "NTFS Volume Serial Number" which seams to match the UUID under Linux.

Possibilities to get the "NTFS Volume Serial" from **Windows**:

- commandline since XP: *fsutil.exe fsinfo ntfsinfo C:*
- under c++

```
HANDLE fileHandle = CreateFile(L"\\\\.\\C:", // or
\Volume{GUID}"
                               GENERIC_READ,
                               FILE_SHARE_READ|FIL
                               NULL,
                               OPEN_EXISTING,
                               NULL,
                               NULL);
DWORD i;
NTFS_VOLUME_DATA_BUFFER ntfsInfo;
DeviceIoControl(fileHandle,
                FSCTL_GET_NTFS_VOLUME_DATA,
                NULL,
                0,
                &ntfsInfo,
                sizeof(ntfsInfo),
                &i,
                NULL));
cout << "UUID is " << std::hex << ntfsInfo.VolumeS
std::hex << ntfsInfo.VolumeSerialNumber.LowPart <<
```

Possibilities to get the UUID under **Linux**:

- ls -l /dev/disk/by-uuid
- ls -l /dev/disk/by-label
- blkid /dev/sda1

Share Improve this answer

Follow

answered Feb 3, 2010 at 18:13

Bernhard

1