The Old New Thing

You can create an infinitely recursive directory tree

27 Dec 2004 7:00 AM | 40

It is possible to create an infinitely recursive directory tree. This throws many recursive directory-traversal functions into disarray. Here's how you do it. (Note: Requires NTFS.)

Create a directory in the root of your C: drive, call it C:\C, for lack of a more creative name. Right-click My Computer and select Manage. click on the Disk Management snap-in.

From the Disk Management snap-in, right-click the C drive and select "Change Drive Letter and Paths...".

From the "Change Drive Letter and Paths for C:" dialog, click "Add", then where it says "Mount in the following empty NTFS folder", enter "C:\C". Click OK.

Congratulations, you just created an infinitely recursive directory.

```
C:\> dir

 Volume in drive has no label
 Volume Serial Number is A035-E01D

 Directory of C:\

08/19/2001  08:43 PM                 0 AUTOEXEC.BAT
12/23/2004  09:43 PM    <JUNCTION>     C
05/05/2001  04:09 PM                 0 CONFIG.SYS
12/16/2001  04:34 PM    <DIR>          Documents and Settings
08/10/2004  12:00 AM    <DIR>          Program Files
08/28/2004  01:08 PM    <DIR>          WINDOWS
               2 File(s)              0 bytes
               4 Dir(s)   2,602,899,968 bytes free

C:\> dir C:\C

 Volume in drive has no label
 Volume Serial Number is A035-E01D

 Directory of C:\C

08/19/2001  08:43 PM                 0 AUTOEXEC.BAT
12/23/2004  09:43 PM    <JUNCTION>     C
05/05/2001  04:09 PM                 0 CONFIG.SYS
12/16/2001  04:34 PM    <DIR>          Documents and Settings
08/10/2004  12:00 AM    <DIR>          Program Files
08/28/2004  01:08 PM    <DIR>          WINDOWS
               2 File(s)              0 bytes
               4 Dir(s)   2,602,899,968 bytes free


C:\> dir C:\C\C\C\C\C\C
```

```
 Volume in drive has no label
 Volume Serial Number is A035-E01D

 Directory of C:\C\C\C\C\C\C

08/19/2001  08:43 PM                 0 AUTOEXEC.BAT
12/23/2004  09:43 PM    <JUNCTION>     C
05/05/2001  04:09 PM                 0 CONFIG.SYS
12/16/2001  04:34 PM    <DIR>          Documents and Settings
08/10/2004  12:00 AM    <DIR>          Program Files
08/28/2004  01:08 PM    <DIR>          WINDOWS
               2 File(s)              0 bytes
               4 Dir(s)   2,602,899,968 bytes free
```

Go ahead and add as many "\C"s as you like. You'll just get your own C drive back again.

Okay, now that you've had your fun, go back to the "Change Drive Letter and Paths for C:" dialog and Remove the "C:\C" entry. Do this before you create some real havoc.

Now imagine what happens if you had tried a recursive treecopy from that mysterious C:\C directory. Or if you ran a program that did some sort of recursive operation starting from C:\C, like, say, trying to add up the sizes of all the files in it.

If you're writing such a program, you need to be aware of reparse points (that thing that shows up as `<JUNCTION>` in the directory listing). You can identify them because their file attributes include the `FILE_ATTRIBUTE_REPARSE_POINT` flag. Of course, what you do when you find one of these is up to you. I'm just warning you that these strange things exist and if you aren't careful, your program can go into an infinite loop.

Blog - Comment List MSDN TechNet

Comments

Chris Lundie
27 Dec 2004 7:16 AM
#
Sure enough, I've written an app that will loop forever and ever when it encounters one of these. It's on a background thread and you can cancel it, but it will never get any real work done.

Stuart Brockman
27 Dec 2004 7:29 AM
#
Probably the best thing to do would be to totally ignore folders with FILE_ATTRIBUTE_REPARSE_POINT set...

Jonathan
27 Dec 2004 7:31 AM
#
Hehe. Time for me to make some changes to one of my programs.... :-)

Diego
27 Dec 2004 7:38 AM
#
Does NTFS support unix-like symbolick links? (It's very easy to create recursive directories in unix, just link to yourself)

I though NTFS supported "hard links" but I'm not sure about "symbolic links". Any pointers?

Elliot
27 Dec 2004 8:29 AM
#
Nice article! I notice that after recursing 16 times (either from cmd.exe or Explorer), I see an error "The name of the file cannot be resolved by the system". Where is that safeguard implemented? "Inside Windows 2000" says that Explorer will only browse 32 directories deep, and cmd.exe will only browse paths less than 256 characters, but this seems to be a third safety net built in at a lower level to prevent misbehaved applications from hurting everyone.

Also, it seems like applications that process directory trees could easily detect if any recursive directories exist in the tree they are about to process by making use of the Win32 mount point scanning functions. For a given volume, scan all the mount points, saving away the source path and checking to see if the target path exists in the list of source paths. Any target path that is also contained in a source path is recursive. Using the mount point enumeration functions along with an STL set makes this dead simple, and gives your application much more sophisticated functionality than simply skipping all reparse points as suggested by a previous comment.

David Phillips
27 Dec 2004 8:33 AM
#
Diego: Junctions are very similar to symbolic links, but they can only be used with directories.

http://www.sysinternals.com/ntw2k/source/misc.shtml#junction

Anonymous Coward
27 Dec 2004 8:54 AM
#
More troubling is that you only have 2GB of disk space left. That is nowhere near enough to swing a computer cat these days :-)

Merle
27 Dec 2004 9:03 AM
#
Hey! I only have ~200M left on each of my three partitions. It's enough if you're careful.

It also helps to be running way older OSes. ;-)

Raymond Chen
27 Dec 2004 9:07 AM
#
Elliot: Watch out for mutual recursion, and good luck getting this to work on a network volume...

AT
27 Dec 2004 9:13 AM
#
Raymond,

I've observed this very-very long time ago.
Mark Rusinovich in his initial version (1.01 or something like this) of junction tool created a link to "C:\Folder\" with slash at end. This disallowed to create junction on existing juction - only on real folder.

I've found that this is trivialy possible to remove a trailing slash and allow creation of junction on junctions.
I.e.
C:\Folder Real rolder
C:\Link1 Link to Folder
C:\Link2 Now possible to make a link to Link1 - not only on real folder

I've emailed my finding to Mark and he included my lame C++ hack into his source code at Junction version 1.02

You can still see my (recently a little bit modified to use tchar.h) 4 lines of lame source code removing trailing slash by zeroing it out - for all cases except "<drive>:\" in current Junction.

In the same time I've found that NT filesystem driver will follow reparse points at most 16 times.
I.e. in case if you do Link17 -> Link16 -> Link15 -> ... -> Link2 -> Link1 -> RealFolder - this will fail.
As well C:\C\C\C\C\C\C\C\C\C\C\C\C will have same 16 limit as Elliot has found.
This is probably to detect infinite loops in reparse links - as I've found this is possible to make a link on link and filesystem will follow them.

Even more - your statement about "infinitely recursive directory tree" is not true due to additional limitation - MAX_PATH length. For some API this is 255 characters, for native - this is 32K (http://msdn.microsoft.com/library/en-us/fileio/base/naming_a_file.asp)
Correct me if I wrong - this is related to unsigned 16-bit Length field in UNICODE_STRING structure.

So - this is not an infinite - but limited by some exponent (something I agree - value of this exponent can be more that number of particles in universe ;-)

Mike Dunn
27 Dec 2004 9:47 AM
#

Diego> NTFS has always supported hard links, but before Win 2K there wasn't an easy way to make one. Win 2K added the CreateHardLink() function.

Peter da Silva
27 Dec 2004 10:13 AM
#
If your program descends the directory tree by changing its current directory each step of the way, won't it avoid blowing out the path length limits?

Brent Dax
27 Dec 2004 10:53 AM
#
Of course, most Unix tools worth the hard drive space they're stored on already know about links--hard and symbolic--and are written to handle them...

"Those who don't understand UNIX are doomed to reinvent it, poorly."
--Henry Spencer

Tim Smith
27 Dec 2004 11:22 AM
#
"Of course, most Unix tools worth the hard drive space they're stored on already know about links--hard and symbolic--and are written to handle them... "

Easy thing to say when you restrict the set of tools so heavily.

asdf
27 Dec 2004 11:22 AM
#
What I've been doing is storing dwVolumeSerialNumber, nFileIndexHigh, and nFileIndexLow (from BY_HANDLE_FILE_INFORMATION) of all directories I process to make sure I don't revisit them. Too lazy to test it on junctions or over the network (it's for an internal tool used for mapping permutations of the same pathname to a unique id, not to protect against infinite recursion anyway). I don't really like skipping over (sym|hard)links as opposed to processing them once, but that's just me.

foxyshadis
27 Dec 2004 12:01 PM
#
I wish PHP and Perl had better native support of NTFS filesystem gotchas like this (and reiserFS and advFS as well), at the risk of bloating them even more. (I wouldn't mind a mod_ntfs.) Their strength is that they can do many formerly complicated things with ease and efficiency, yet their only support of remotely advanced filesystems is through raw system calls - thankfully PHP5's win32 api finally _works_.

I would be so glad to leave vbscript behind for specialized windows tasks if I could find anything else that didn't feel like writing a compiler in VB3.

Peter, if you change to the folder, such as cd C:\C\C\C, you'll find yourself in a folder with that label - that is, after all, the expected behavior. You're trying to trick programs into believing that some other folder is really here, for some convenience.

Joe Dietz
27 Dec 2004 12:08 PM
#
Having actually implemented reparse point support in an IFS file system before, I found somewhat different behavior in the server versions of the OS. I though that reparse point recursion was limitted to some fairly small number like 8 STATUS_REPARSE_POINT returns from the file system. Something about how DFS is structured with (8?) levels of nesting when using reparse points also led me to believe this.

Anyways I added my own recusion checking into my reparse point implementation just to be safe so as to not allow such things from happening.

Memet
27 Dec 2004 12:39 PM
#
AT: Thanks for that info man, it's exactly what I needed to know. I'm wondering if there is some other hack (a-la inheritable permissions etc) that would allow me to limit to a single recurse down.
I have a website engine for multiple users that can be accessed via either user.example.com or www.example.com/~user. The way it works is that when a user site is created, the app just symlinks the root folder to itself using ~user. It works great, except that you can write garbage like www.example.com/~user1/~user2/~user22314...
Which is fine, the internal logic recognizes garbage and just returns "site not found", but I was curious if there's a way to limit the recursion depth to a single level.

Anyways, happy new year to you all.

pompo500
27 Dec 2004 12:44 PM
#
These are the kind of articles I visit your blog for. Interesting, educating and fun. More of these please!

Memet
27 Dec 2004 12:49 PM
#
The recursion depth, quite interestingly, is not limited to the number of junctions followed, but rather to the number of directories in the entire path... at least that's the way the command prompt behaves (maybe it's a prompt limitation, I haven't tested direct API calls).
So if you start this self recursion at say "z:\data\folder\~folder", then you get only 14 junction recursions.
More interesting: if you create a 16 folder deep directory structure and create a junction

at the deepest folder, you can still delve down. But after a while (which I couldn't figure out the logic for) cmd.exe returns:
"The name of the file cannot be resolved by the system."

It seems to me like there was a manual hack added into the NTFS driver that maybe doesn't cover all cases?

Memet
27 Dec 2004 1:06 PM
#
Joe Dietz: did you detect cycles? or did you check if two adjacent symlinks where the same?
For example, you could have this1 and this2 point to the .\
Would your driver detect .\this1\this2\this1\this2...? (just checking =)

Brandon Paddock
27 Dec 2004 1:39 PM
#
"Probably the best thing to do would be to totally ignore folders with FILE_ATTRIBUTE_REPARSE_POINT set... "

And this is why mounting drives as folders doesn't work as expected for some of us =/

Peter da Silva
27 Dec 2004 3:33 PM
#
foxyshadis: the label on the directory is only relevant if the program generates a fully qualified path name for the files or directories it's traversing.

If, for example, it's just looking for files of a certain kind or containing certain information (for example, it's looking for files over a certain size, or files indicating that an index is to be created for the directory on a website, or just pulling some information out of configuration files) then it may get in the loop without ever attempting to pass an excessively long file name or path to the OS.

Surf.
27 Dec 2004 6:33 PM
#
Will it end up in a stack overflow?

Norman Diamond
27 Dec 2004 7:33 PM
#
I once had to fix up a partition that had an infinitely recursive directory tree. It was on FAT (16 or 32 I don't recall) under Windows 95 B. I do not know what my then-boss had done to create it. Windows 95 Explorer and DOS command prompt were all following it as

they would any normal directory entry, same as VMS did with its 000000 directory entries. I did get rid of it, but don't remember how.

Ian Argent
27 Dec 2004 7:49 PM
#
Is there any way for me to mount a UNC name (on a network server, not local) so that it appears to be a local directory? Specifically, I need to point \documents & Settings\<username>\My documents to \\servename\sharename\documents and have this be invisible to a cranky application. (\\servername\sharename\documents resides on a RAID, but I have a couple of applications that take various forms of exception to "My Documents" beings mapped to a UNC or to a mapped network drive - those being the options I tried)

余啊雷
27 Dec 2004 8:42 PM
#
lazybones &raquo;

:-(
28 Dec 2004 8:27 AM
#
In real applications it seems that nobody cares about FILE_ATTRIBUTE_REPARSE_POINT flag. I tested some applications like AntiVir and Nero Burning ROM and they all do the recursion.

Joe Dietz
28 Dec 2004 9:00 AM
#
Memet: when writing a file system you are in control of the verticaland the horizontal. Its actually fairly easy to detect recursion by simply detecting reentrancy. When you raise status reparse you increment a field of your irp context structure. If that counter exceeds whatever recusion thresh hold you choose you don't raise status reparse, you raise a invalid path status instead. I actually did this based on what I believe to be NTFS's reparse point behaviors at least on the server versions of the OS. Perhaps I was mistaken about this observation due to all of the reports here to the contrary. But it is not clear if we are all talking about the same apples here.

Path length checks at least for things like cmd.exe and explorer.exe are all done in the win32 layer btw. This seems to be for 16-bit compatibility reasons. NTFS and most other file systems have max path checking, as well, but typically on whole integer sized paths (the unicode path length is how this is experienced by win32 apps since all file systems see UTF paths that win32 generates or passes through to the kernel).

Matt
28 Dec 2004 11:48 AM

#
Norman, I'll up your ante: I remember playing around with this type of thing on the Commodore 64 disk drive to 'hide' files. The file structure was a linnked list and you could edit the link to point back to itself.

Man, I was a geek even back in middle school...

余啊雷
29 Dec 2004 10:00 AM
#
The problem with enumeration is that somebody always loses.

Joshua Schaeffer
30 Dec 2004 4:24 PM
#
I remember deleting a reparse point from within Windows 2000's Explorer and rather than simply kill the link, it delved inside and started recursively deleting the guts of the linked folder. I was just a trifle disgusted.

Peter da Silva
1 Jan 2005 11:22 AM
#
"(set 'yak '(yak)) (rplacd yak yak) yak" -- Lisp 1.5, 1970's

Stefan Kanthak
1 Jan 2005 9:15 PM
#
Joshua: that's one of the nasty sides of Explorer!
If you want to use junctions with the GUI you'd better install a shell extension like "NTFS Link" from http://www.elsdoerfer.net/ which implements the necessary hooks and handlers to repair these flaws.

余啊雷
4 Jan 2005 12:06 AM
#

余啊雷
10 Jan 2005 7:45 PM
#

余啊雷
1 Jun 2006 6:15 AM
#

In my previous post I explained how you could build a class that uses recursion to scan through directories....

余啊雷
25 Oct 2006 10:00 AM
#

Punt.

余啊雷
26 Oct 2006 3:42 AM
#

PingBack from http://rohand.com/2006/10/26/extremely-important-no-joke/

余啊雷
2 Jun 2009 11:16 AM
#

Shows how to download files from a remote FTP server. Demonstrates how to handle files which already exist on the target disk, how to work with symlinks and hardlinks. Shows how to let the user choose what to do when a transfer problem is detected.