The Old New Thing

What is the programmatic equivalent to unchecking the box to prevent a file from having its contents indexed?

20 Feb 2014 7:00 AM | 25

A customer wanted to know how they could write a program that automatically checked and unchecked the box that appears on a file's property sheet on the *General* tab, clicking the *Advanced* button, and then checking or unchecking the item whose name keeps changing:

- On Windows 7, it's called *Allow this file to have contents indexed in addition to file properties*.
- On Windows Vista, it's called *Index this file for faster searching*.
- On Windows 2000 and Windows XP, it's called *For fast searching, allow Indexing Service to index this folder*.

The checkbox maps to the file attribute formally known as `FILE_ATTRIBUTE_NOT_CONTENT_INDEXED`, and informally known as `FANCI` (pronounced like the word *fancy*). Checking the box clears the attribute, and unchecking the box sets the attribute.

The customer liaison replied, "Thanks for your assistance. The customer was able to use the `System.IO.File.SetAttributes` method with the values in the `System.IO.FileAttributes` enumeration to manipulate the indexing attribute. The customer has a follow-up question, however: 'I need this change to be applied recursively to all files in a entire directory subtree. Is there a single series of Visual Basic commands that will accomplish this, or do I need to write a loop by hand?'"

This question kind of ran off topic for the product team and fell more in line with Developer Support. I suggested that the follow-up question be redirected to the Visual Basic support team.

For me, it was interesting that (1) the customer liaison was himself not aware enough to realize that the question had changed audiences, and (2) the customer was so helpless that they couldn't look for the answer themselves.

Bonus chatter: The file system itself pays no attention to the FANCI bit. It's just a signal to any program that does file content indexing that this particular file should be skipped. Whether any particular search index program respects the flag is something you need to take up with the vendor of that search index program. (As far as I'm aware, all Microsoft search indexes should honor the flag.)

Blog - Comment List MSDN TechNet

Comments

AC
20 Feb 2014 7:17 AM
#

For me, it's interesting how different the levels of "advancedness" of the questions are.

The first question seems very legitimate. I would have thought someone asking it is way beyond the level needed to figure out the second question.

Brian_EE
20 Feb 2014 7:23 AM
#

or (3) the customer was just lazy, and thought that since they had received such a helpful answer the first time, they would just ask the next question instead of looking for it themselves.

Joshua
20 Feb 2014 7:28 AM
#

I think (4) non programming sysadmin.

alegr1
20 Feb 2014 7:41 AM
#

@Joshua:

non programming sysadmin would do:

attrib -i /s

Joshua
20 Feb 2014 8:19 AM
#

Man that makes the second question so easy.

Shell "attrib -i /s " + path

John
20 Feb 2014 8:47 AM
#

These types of questions come up on StackOverflow all the time, if he's using .NET 4 or above the Directory.EnumerateFiles(string,string,SearchOption.AllDirectories) is his friend.

I've found myself using this pattern on more than one occasion in this manner:

// Grab the files and perform the action in parallel.

var targetFiles = Directory.EnumerateFiles(targetPath, "*.cs",
SearchOption.AllDirectories);

```
Parallel.ForEach(
  targetFiles,
  targetFile =>
  {
     // Perform some Action
  });
```

Obviously you can get a bunch more complex but take what you will from there. If I ever meet the team that did the Parallel Task Library I'll have to buy them a case of beer (or soda).

tocsa
20 Feb 2014 9:08 AM
#

"The file system itself pays no attention to the FANCI bit. It's just a signal to any program that does file content indexing that this particular file should be skipped."

Does Windows has indexing service tho which would consume less resources? I uncheck that after every clean install recursively for the whole drive.

Bonus off-topic question: Is there any trick for speed up NTFS file systems like "noatime" mount option on Linux?

Mark
20 Feb 2014 9:16 AM
#

tocsa: noatime was enabled in Vista by default

Also, you can just disable the Windows Indexing service or remove paths from it, instead of unticking the "include in index" checkbox.

Rick C
20 Feb 2014 9:23 AM
#

@tocsa, see blogs.msdn.com/.../10222560.aspx: "Starting in Windows Vista, maintaining the last-access time is disabled by default."

Chris Crowther
20 Feb 2014 11:09 AM
#

I would guess the customer wasn't actually a developer. Depending on the size of company the customer works for it's possible they're just a general purpose IT bod who got tasked to do something for some esoteric reason.

xpclient
20 Feb 2014 11:21 AM
#

Well thankfully Windows Search indexer respects it. Like countless other broken things which are By Design, I wouldn't be surprised if the legacy Indexing Service had been the last to respect it. One of those excuses for 'mating call of the loser'.

> ["*Thankfully it works, like everything else that's broken." I can't make any sense of this, but please don't try to explain. -Raymond*]

bmm6o
20 Feb 2014 11:47 AM
#

What I want to know is how these customers get their (frequently) dumb questions to Raymond in the first place. It costs me $80 to even open a support case with Microsoft, and this guy is asking if he needs to write a loop? Or the question from 3 weeks ago (blogs.msdn.com/.../10495426.aspx) where the guy asks about large pids - I assume he didn't have to pay to ask that question. What other kinds of support arrangements exist that make these questions reasonable (from the perspective of the asker)?

Jason T. Miller
20 Feb 2014 12:07 PM
#

@bmm6o: programs like MSDN bundle a set number of no-charge, "use 'em or lose 'em" support incidents per year. Also, at most companies, developers and sysadmins don't pay their own support bills.

Mr Cranky
20 Feb 2014 1:56 PM
#

Holy double negative, Batman!

Really? You *check* a box to *clear* an attribute; an attribute that has the word *NOT" in it.

Are there normal humans who can parse through that? I'd just have to wing it, and see what happens.

Matt
20 Feb 2014 4:07 PM
#

@John:

Why are you doing I/O bounded tasks in a Parallel.ForEach?

cheong00
20 Feb 2014 10:25 PM
#

For (1): The (non-senior) CS staffs' problem solving skills are usually limited to the predefined scripts given.

For (2): "Afterall the company has paid for the support, why not make the maximum use of it?"

So hardly a surprise at all. :P

Azarien
21 Feb 2014 1:45 AM
#

@Mr Cranky: what's so hard? The checkbox says "Allow this file to have contents indexed", and the attribute is FILE_ATTRIBUTE_NOT_CONTENT_INDEXED.

So:

checkbox checked => attribute cleared.

checkbox unchecked => attibute set.

there's no double negative, it's single negative.

Sven2
21 Feb 2014 3:04 AM
#

I don't think the second question was inappropriate at all. The customer didn't need help in "writing the loop", but just wanted to know if a loop is necessery.

It could be that there's a command like SetAttributesRecursively or that SetAttributes can be used to apply attribute changes recursively by calling it in a special syntax. A possible advantage over writing the loop manually would be that such a command could be issued atomically (so you don't end up with half of the files indexed if your program is killed during the loop).

> [*The point is that they directed the follow-up question to the wrong team. The original question was a file system question; the follow-up was a VB question. -*

Raymond]

Alex Cohn
21 Feb 2014 6:24 AM
#

The followup question could be legitimate: "can I set the flag once to exclude the bazillion files that will be created in my directory"? The answer does not require MS tech support, only one search, and it is: [no](books.google.co.il/books).

John
21 Feb 2014 7:21 AM
#

@Matt:

Because you maximize your throughput ensuring that the limiting factor really is I/O.

Is there a reason why you wouldn't throw this in a Parallel Loop? It also offers an extensibility point for doing far more complex tasks such as parsing the input and acting upon its results; but I'm always open to a different view point.

Bob
21 Feb 2014 7:49 AM
#

@John

>Is there a reason why you wouldn't throw this in a Parallel Loop?

Because you run the risk of (drastically) reducing performance by causing the disk head to thrash around. May or may not be an issue in this case, but something to keep in mind whenever dealing with hardware that's not constant time random access.

John
21 Feb 2014 8:32 AM
#

@Bob/Matt:

I was interested in your comments I wrote the following test program to validate my assumptions (please forgive the formatting):

```
static void Main(string[] args)

{

    string targetPath = @"R:\TestMultiThreaded";
```

```
Environment.SetEnvironmentVariable("TMP", targetPath);
//for (int i = 0; i < 65535; i++)
//{
//    string tempFilePath = Path.GetTempFileName();
//}
var targetFiles = Directory.EnumerateFiles(targetPath);
int numberOfSamples = 5;
for (int i = 0; i < numberOfSamples; i++)
{
    Stopwatch walltime = Stopwatch.StartNew();
    foreach (string targetFile in targetFiles)
    {
        FileInfo fi = new FileInfo(targetFile);
        fi.Attributes = fi.Attributes | FileAttributes.NotContentIndexed;
    }
    walltime.Stop();
    Console.WriteLine("Sequential Operation: {0}", walltime.Elapsed.ToString());
}
for (int i = 0; i < numberOfSamples; i++)
{
    Stopwatch walltime = Stopwatch.StartNew();
    Parallel.ForEach(
        targetFiles,
        targetFile =>
        {
            FileInfo fi = new FileInfo(targetFile);
            fi.Attributes = fi.Attributes | FileAttributes.NotContentIndexed;
        });
    walltime.Stop();
    Console.WriteLine("Parallel Operation: {0}", walltime.Elapsed.ToString());
}
```

```
}
```

Returned the following results:

Sequential Operation: 00:00:06.1964492

Sequential Operation: 00:00:06.1566922

Sequential Operation: 00:00:06.1519958

Sequential Operation: 00:00:06.1584689

Sequential Operation: 00:00:06.1496031

Parallel Operation: 00:00:01.9879020

Parallel Operation: 00:00:01.9605536

Parallel Operation: 00:00:01.9339526

Parallel Operation: 00:00:01.9310271

Parallel Operation: 00:00:01.9494726

This was tested on HP Compaq 8200 Elite that is equipped with a Core i7 2600 @ 3.40GHz with 20480 MBytes of RAM. The Target Drive is a 4GB RAM Disk created using IMDisk (The RAM is some cheap Mushkin PC3-10700). This is a standard development PC here (although now most come with 24GB and a 240GB SSD and a much more current i7).

The first part of the program creates 65535 zero byte Temporary files on the drive, these files are then modified to have their attributes have the NotContentIndex flag applied to them sequentially via a ForEach Loop, executed 5 times and then again in Parallel 5 times. The amount of time required to process this operation is cut to a third of the time with little readability changes.

I understand the argument of disk thrashing but isn't SuperFetch supposed to solve most of these issues in the brave new world of Vista+? Of course being on a RAM drive this was taken out of this equation.

> [*Congratulations, you demonstrated that if there is no I/O latency, then I/O latency is not a bottleneck. -Raymond*]

John
21 Feb 2014 9:40 AM
#

Is there a better way to test this? I pointed the program at my Mechanical Drive (Hitachi HDS721010CLA332) and came up with the following results:

Sequential Operation: 00:00:06.9819641

Sequential Operation: 00:00:07.8233557

Sequential Operation: 00:00:07.0141172

Sequential Operation: 00:00:07.9264859

Sequential Operation: 00:00:07.2037502

Parallel Operation: 00:00:05.5713380

Parallel Operation: 00:00:02.5200298

Parallel Operation: 00:00:04.1874065

Parallel Operation: 00:00:05.0852479

Parallel Operation: 00:00:02.4919081

Obviously slower once you're not on a RAM Drive. The sequential operation providing a more consistent time as compared to the Parallel Operation (while faster in all cases had a greater variance).

Again obviously this test is not without flaw, what is a more accurate way to test? How do you test the possibility of disk thrashing? How do you go to upper management and convince them that one way is more optimal than the other?

> [*Eventually all the data needed is in cache so you aren't actually hitting the disk any more. You need to test on, say, a removable drive, and unplug it between runs. - Raymond*]

Joker_vD
21 Feb 2014 10:19 AM
#

A "fool your filesystem driver optimization" contest is going on?

John
21 Feb 2014 10:31 AM
#

@Raymond

Thanks! I appreciate your blog and I strive to learn something everyday.