Subject: parse subdirectories
Posted by Helder Marchetto on Mon, 13 Mar 2017 16:32:11 GMT
View Forum Message <> Reply to Message

Hi,
I have widget application that takes a directory as input and generates a widget_tree of the (sub)directory structure.
I've have done this using basically something like (there's more to it that this...):

```
pro dirParser::addsubtree, dir
subs = file_search(dir+'*', /test_directory, count=cnt)
subParent = self.widgets.treeSub[-1]
if cnt gt 0 then foreach sub,subs do self->addSubsTree, sub, subParent
end
```

This works fine, meaning it returns the directory structure and that's great. However, I would like to switch the windows powershell to get the tree structure. Why? Because the above is really slow.

So, I can call the powershell like this:
```
spawn, 'powershell -WindowStyle Hidden "Get-ChildItem -Recurse | ?{ $_.PSIsContainer } | Select-Object FullName"', result, /noshell
```

which gives me a very quick response with something like:
```
IDL> print, transpose(result)

FullName
--------
K:\data\sub-1\2002
K:\data\sub-1\2004
K:\data\sub-1\2005
K:\data\sub-1\2017
K:\data\sub-1\2002\02_01_26
K:\data\sub-1\2002\02_01_28
K:\data\sub-1\2004\04_12_02
K:\data\sub-1\2004\04_12_03
```

and so on (there are many more subdirectories).
Does anybody have a good suggestion how to parse the text contained in the above result array?

I know how to handle strings, but I don't have a good way to sort the subdirectories (for instance "K:\data\sub-1\2002\02_01_26" is a subdirectory of "K:\data\sub-1\2002", but comes only after all the other same level directories are listed).

I would appreciate any suggestion on how to solve the directory listing chaos.

Regards,
Helder

PS: Just for the time comparison: running the above powershell command over a structure of >2000 directories/subdirectories took "just" 8 seconds. The older method took minutes...

Subject: Re: parse subdirectories
Posted by wlandsman on Mon, 13 Mar 2017 17:41:21 GMT
View Forum Message <> Reply to Message

I don't have an answer to your question, but do want to point out that the slowness of FILE_SEARCH() on Windows is a long-standing problem.

http://www.idlcoyote.com/code_tips/fastsearch.php

and one that I've ranted about before

https://groups.google.com/forum/#!search/file_search$20wayne/comp.lang.idl-pvwave/ABttLQ5NHHU/xgSFcodk4N0J

And just yesterday I became aware of another group at Goddard encountering this problem. Please Harris/Exelis can we get a fix for FILE_SEARCH?

Anyway, for a recursive directory search, I'm not sure that FINDFILE() or David Fanning's listfile.pro are a suitable solution. I suspect -- but am not certain -- that you do have to sort the output of the powershell as you are doing. You might have to count the number of backslashes to get the directory level. --Wayne

On Monday, March 13, 2017 at 12:32:12 PM UTC-4, Helder wrote:
> Hi,
> I have widget application that takes a directory as input and generates a widget_tree of the (sub)directory structure.
> I've have done this using basically something like (there's more to it that this...):
>
> pro dirParser::addsubtree, dir
> subs = file_search(dir+'*', /test_directory, count=cnt)
> subParent = self.widgets.treeSub[-1]
> if cnt gt 0 then foreach sub,subs do self->addSubsTree, sub, subParent
> end
>
> This works fine, meaning it returns the directory structure and that's great. However, I would like to switch the windows powershell to get the tree structure. Why? Because the above is really slow.
>
> So, I can call the powershell like this:
> spawn, 'powershell -WindowStyle Hidden "Get-ChildItem -Recurse | ?{ $_.PSIsContainer } | Select-Object FullName"', result, /noshell

>
> which gives me a very quick response with something like:
> IDL> print, transpose(result)
>
> FullName
> --------
> K:\data\sub-1\2002
> K:\data\sub-1\2004
> K:\data\sub-1\2005
> K:\data\sub-1\2017
> K:\data\sub-1\2002\02_01_26
> K:\data\sub-1\2002\02_01_28
> K:\data\sub-1\2004\04_12_02
> K:\data\sub-1\2004\04_12_03
>
> and so on (there are many more subdirectories).
> Does anybody have a good suggestion how to parse the text contained in the above result array?
>
> I know how to handle strings, but I don't have a good way to sort the subdirectories (for instance "K:\data\sub-1\2002\02_01_26" is a subdirectory of "K:\data\sub-1\2002", but comes only after all the other same level directories are listed).
>
> I would appreciate any suggestion on how to solve the directory listing chaos.
>
> Regards,
> Helder
>
> PS: Just for the time comparison: running the above powershell command over a structure of >2000 directories/subdirectories took "just" 8 seconds. The older method took minutes...

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Mon, 13 Mar 2017 23:04:14 GMT
View Forum Message <> Reply to Message

Thanks for the insight Wayne.
I think I will go for the sort/count slashes option. Otherwise programming is no fun.
Cheers, Helder

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Wed, 15 Mar 2017 11:06:18 GMT
View Forum Message <> Reply to Message

On Tuesday, March 14, 2017 at 12:04:15 AM UTC+1, Helder wrote:
> Thanks for the insight Wayne.

> I think I will go for the sort/count slashes option. Otherwise programming is no fun.
> Cheers, Helder

Hi,
I was just about to give up on this because I noticed that if the output directory is too long, Windows shortens the directories and adds three dots. Something like this:
K:\firstDir\secondDir\thirdDir\four...
instead of
K:\firstDir\secondDir\thirdDir\fourthDir\

This is literaly what is returned by the spawn command:
spawn, 'powershell -WindowStyle Hidden "Get-ChildItem -Recurse | ?{ $_.PSIsContainer } | Select-Object FullName"', result, /noshell

After fiddling around with cmd and powershell, I found finally the answer from stackoverflow:
http://stackoverflow.com/questions/9528039/powershell-why-does-out-file-break-long-line-into-smaller-lines

So this now works better:
spawn, 'powershell -WindowStyle Hidden "Get-ChildItem -Recurse | ?{ $_.PSIsContainer } | Select-Object FullName | Format-list *"', result, /noShell

There is still however a little problem with overflowing lines. This is simply dealt with by searching for those lines that do not contain ":" and moving the content to the previous line...

Cheers,
Helder

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Wed, 15 Mar 2017 15:02:39 GMT
View Forum Message <> Reply to Message

On Tuesday, March 14, 2017 at 12:04:15 AM UTC+1, Helder wrote:
> Thanks for the insight Wayne.
> I think I will go for the sort/count slashes option. Otherwise programming is no fun.
> Cheers, Helder

Ok,
so I managed this and it is working fine. The improvement is there: crawling through 2000 sub-directories takes ~9 seconds against the >14 sec with the old file_search method.
Now the time to execute is limited by the spawn command (8.5 sec), whereas before it was limited by the file_search (13 sec).
If anybody requests it and I have time I'll put the code together in a "nicer" way.
Cheers,
Helder

Subject: Re: parse subdirectories
Posted by lecacheux.alain on Wed, 15 Mar 2017 18:23:44 GMT
View Forum Message <> Reply to Message

Le mercredi 15 mars 2017 16:02:42 UTC+1, Helder a écrit :
> On Tuesday, March 14, 2017 at 12:04:15 AM UTC+1, Helder wrote:
>> Thanks for the insight Wayne.
>> I think I will go for the sort/count slashes option. Otherwise programming is no fun.
>> Cheers, Helder
>
> Ok,
> so I managed this and it is working fine. The improvement is there: crawling through 2000 sub-directories takes ~9 seconds against the >14 sec with the old file_search method.
> Now the time to execute is limited by the spawn command (8.5 sec), whereas before it was limited by the file_search (13 sec).
> If anybody requests it and I have time I'll put the code together in a "nicer" way.
> Cheers,
> Helder

Maybe you could simply reorder your string array first by using SORT ?

print, result[sort(result)]

K:\data\sub-1\2002
K:\data\sub-1\2002\02_01_26
K:\data\sub-1\2002\02_01_28
K:\data\sub-1\2004
K:\data\sub-1\2004\04_12_02
K:\data\sub-1\2004\04_12_03
K:\data\sub-1\2005
K:\data\sub-1\2017

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Wed, 15 Mar 2017 20:05:22 GMT
View Forum Message <> Reply to Message

On Wednesday, March 15, 2017 at 7:23:46 PM UTC+1, alx wrote:
> Le mercredi 15 mars 2017 16:02:42 UTC+1, Helder a écrit :
>> On Tuesday, March 14, 2017 at 12:04:15 AM UTC+1, Helder wrote:
>>> Thanks for the insight Wayne.
>>> I think I will go for the sort/count slashes option. Otherwise programming is no fun.
>>> Cheers, Helder
>>
>> Ok,
>> so I managed this and it is working fine. The improvement is there: crawling through 2000 sub-directories takes ~9 seconds against the >14 sec with the old file_search method.
>> Now the time to execute is limited by the spawn command (8.5 sec), whereas before it was

limited by the file_search (13 sec).
>> If anybody requests it and I have time I'll put the code together in a "nicer" way.
>> Cheers,
>> Helder
>
> Maybe you could simply reorder your string array first by using SORT ?
>
> print, result[sort(result)]
>
> K:\data\sub-1\2002
> K:\data\sub-1\2002\02_01_26
> K:\data\sub-1\2002\02_01_28
> K:\data\sub-1\2004
> K:\data\sub-1\2004\04_12_02
> K:\data\sub-1\2004\04_12_03
> K:\data\sub-1\2005
> K:\data\sub-1\2017

Hi Alx,
that's indeed *one* of the things I have to do. However, there's more to it than plain sorting. This because I want to maintain the physical tree structure: therefore I have to determine if any given directory is a node or a leaf and index it accordingly.
Since I like to use this a lot - and wonder why this isn't already available - I will try to order/clean things up and share it when it's done.

What it does not do and will no do, is follow symbolic links and I will not test this on a linux/mac machine (simply don't have one). If anybody is interested, we can share the load :-)

Cheers,
Helder

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Thu, 16 Mar 2017 12:29:18 GMT
View Forum Message <> Reply to Message

On Wednesday, March 15, 2017 at 9:05:25 PM UTC+1, Helder wrote:
> On Wednesday, March 15, 2017 at 7:23:46 PM UTC+1, alx wrote:
>> Le mercredi 15 mars 2017 16:02:42 UTC+1, Helder a écrit :
>>> On Tuesday, March 14, 2017 at 12:04:15 AM UTC+1, Helder wrote:
>>>> Thanks for the insight Wayne.
>>>> I think I will go for the sort/count slashes option. Otherwise programming is no fun.
>>>> Cheers, Helder
>>>
>>> Ok,
>>> so I managed this and it is working fine. The improvement is there: crawling through 2000 sub-directories takes ~9 seconds against the >14 sec with the old file_search method.
>>> Now the time to execute is limited by the spawn command (8.5 sec), whereas before it was

limited by the file_search (13 sec).
>>> If anybody requests it and I have time I'll put the code together in a "nicer" way.
>>> Cheers,
>>> Helder
>>
>> Maybe you could simply reorder your string array first by using SORT ?
>>
>> print, result[sort(result)]
>>
>> K:\data\sub-1\2002
>> K:\data\sub-1\2002\02_01_26
>> K:\data\sub-1\2002\02_01_28
>> K:\data\sub-1\2004
>> K:\data\sub-1\2004\04_12_02
>> K:\data\sub-1\2004\04_12_03
>> K:\data\sub-1\2005
>> K:\data\sub-1\2017
>
> Hi Alx,
> that's indeed *one* of the things I have to do. However, there's more to it than plain sorting. This because I want to maintain the physical tree structure: therefore I have to determine if any given directory is a node or a leaf and index it accordingly.
> Since I like to use this a lot - and wonder why this isn't already available - I will try to order/clean things up and share it when it's done.
>
> What it does not do and will no do, is follow symbolic links and I will not test this on a linux/mac machine (simply don't have one). If anybody is interested, we can share the load :-)
>
> Cheers,
> Helder

Ok, so I loaded this up on github with an example:
https://github.com/heldermarchetto/IDL
I'm not that much into github, so if you want to contribute/improve the procedure, you're welcome. Let me know if I can do anything about it.
The notifications are not really set-up and I think I don't need to... I'll check that later.

I had someone test this for me with a much bigger directory tree and the improvement was from 2'17" to 51". Nice.

I think that this procedure will not be useful for small directories! It will probably be slower, and that is because of the spawn call to the powershell.

Cheers,
Helder

Subject: Re: parse subdirectories
Posted by Matthew Argall on Thu, 16 Mar 2017 15:07:55 GMT
View Forum Message <> Reply to Message

Another option would be to populate the widget tree only when the expand node button is clicked. Like an "ls" instead of an "ls -r". This would (I think) involve destroying and regenerating the widget tree each time while updating the widget base via widget_control, [/UPDATE | /REDRAW].

Subject: Re: parse subdirectories
Posted by bradgom on Thu, 16 Mar 2017 17:29:21 GMT
View Forum Message <> Reply to Message

I ran into this issue several years ago while trying to find a way to select files in a tree for quick analysis in IDL. (Boy it would be nice if IDL had a fast file listing function in Windows, and also some built-in file management widgets..) I also don't use GitHub, but I've posted a widget object here:
https://github.com/bradgom/BGDirtree_widget

This doesn't solve the directory crawling time issue, but does only dig down into the currently selected branch instead of the whole directory tree.

Also, since Spawn doesn't work in the VM, this code is still slow when run in VM applications.

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Fri, 17 Mar 2017 08:17:47 GMT
View Forum Message <> Reply to Message

On Thursday, March 16, 2017 at 6:29:25 PM UTC+1, bra...@gmail.com wrote:
> I ran into this issue several years ago while trying to find a way to select files in a tree for quick analysis in IDL. (Boy it would be nice if IDL had a fast file listing function in Windows, and also some built-in file management widgets..) I also don't use GitHub, but I've posted a widget object here:
> https://github.com/bradgom/BGDirtree_widget
>
> This doesn't solve the directory crawling time issue, but does only dig down into the currently selected branch instead of the whole directory tree.
>
> Also, since Spawn doesn't work in the VM, this code is still slow when run in VM applications.

Hi,
great work! I still have to learn how to write compound widgets :-)
I have to point out, that spawn DOES work in the VM.

Cheers,
Helder

Subject: Re: parse subdirectories
Posted by Helder Marchetto on Thu, 18 May 2017 09:09:10 GMT
View Forum Message <> Reply to Message

On Friday, March 17, 2017 at 9:17:49 AM UTC+1, Helder wrote:
> On Thursday, March 16, 2017 at 6:29:25 PM UTC+1, bra...@gmail.com wrote:
>> I ran into this issue several years ago while trying to find a way to select files in a tree for quick analysis in IDL. (Boy it would be nice if IDL had a fast file listing function in Windows, and also some built-in file management widgets..) I also don't use GitHub, but I've posted a widget object here:
>> https://github.com/bradgom/BGDirtree_widget
>>
>> This doesn't solve the directory crawling time issue, but does only dig down into the currently selected branch instead of the whole directory tree.
>>
>> Also, since Spawn doesn't work in the VM, this code is still slow when run in VM applications.
>
> Hi,
> great work! I still have to learn how to write compound widgets :-)
> I have to point out, that spawn DOES work in the VM.
>
> Cheers,
> Helder

Hi,
I've just finished modifying Bradgom's BGDirtree_widget compound widget.
The new version (HMDirTree_Widget) I've created is largely based on his, with some important changes:
1) HMDirTree_Widget does not issue one spawn command per folder. Issuing
   so many spawn commands made the windows task bar go crazy.
2) HMDirTree_Widget does not handle files, only folders.
3) HMDirTree_Widget has no dependencies
4) HMDirTree_Widget always starts by searching for the available fixed
   drives (hard drives and network drives) and creating a list.
   To avoid hanging on slow network connections, the network drives are
   listed, but only explored when the user selects them.
5) HMDirTree_Widget allows only to select single directories, not multiple.
6) HMDirTree_Widget works only with IDL versions 8.0 or higher.
7) HMDirTree_Widget works on Windows only.

If anybody is interested, the code can be found here.
https://github.com/heldermarchetto/HMDirTree_Widget
As I said, it is largely based on Bradgom's work, so thanks to him for the hard work and for making his code available.

Regards,
Helder