Subject: cd to nonexistent directory and up again
Posted by tom.grydeland on Tue, 16 Oct 2012 10:33:18 GMT
View Forum Message <> Reply to Message

Why is it that this fails:

IDL> cd, 'bobo/..'
% CD: Unable to change current directory to bobo/...
  No such file or directory

but this does not:

IDL> cd, './bobo/..'
IDL>

I am using CD to determine whether a given path exists as a directory, and I was expecting the second case to fail just like the first one does, since there is no difference between them in my mind.

Cheers,

Tom

Subject: Re: cd to nonexistent directory and up again
Posted by tom.grydeland on Wed, 17 Oct 2012 13:05:14 GMT
View Forum Message <> Reply to Message

On Tuesday, October 16, 2012 12:33:19 PM UTC+2, tom.gr...@gmail.com wrote:

> I am using CD to determine whether a given path exists as a directory, and I was expecting the second case to fail just like the first one does, since there is no difference between them in my mind.

Thanks to all those who replied, but I am afraid you're not touching upon my question. I am sorry I have not made my point clearly enough.

1) In a world where directories can be symbolically linked, it is not necessarily true that './bobo/...' is equal to '.', even when 'bobo' exists. Truncating away 'bobo/..' on the assumption that they are the same is arguably incorrect, but it is certainly confusing if it is done in some cases and not in others!

2) In the regular Unix shells, 'ls -ld bobo/..' and 'ls -ld ./bobo/..' both fail, with the error message that the file or directory does not exist.

3) FILE_TEST shows the same strange difference between 'bobo/..' and './bobo/..':

```
IDL> print, file_test('./bobo/..', /dir)
           1
IDL> print, file_test('bobo/..', /dir)
           0
```

It looks like I will have to do what I was hoping to avoid:

1) given a path like 'foo/bar/baz/../quux/..', use file_test(x, /dir) on every sub-path x:
  file_test('foo', /dir)
  file_test('foo/bar', /dir)
  file_test('foo/bar/baz', /dir)
  file_test('foo/bar/baz/..', /dir)
  file_test('foo/bar/baz/../quux', /dir)
  file_test('foo/bar/baz/../quux/..', /dir)

If IDL cannot be relied upon to fail in the next step when one or more of these path components don't exist, then I must test every one of them.

2) If all of these tests pass, then

  cd 'foo/bar/baz/../quux/..', curr=old
  cd, old, curr=pwd

3) pwd is now the fully-qualified path to the directory indicated by 'foo/bar/baz/../quux/..'

Cheers,

Tom

Subject: Re: cd to nonexistent directory and up again
Posted by Lajos Foldy on Wed, 17 Oct 2012 15:25:57 GMT
View Forum Message <> Reply to Message

On Wednesday, October 17, 2012 3:05:14 PM UTC+2, tom.gr...@gmail.com wrote:
> On Tuesday, October 16, 2012 12:33:19 PM UTC+2, tom.gr...@gmail.com wrote:
>
>
>
>> I am using CD to determine whether a given path exists as a directory, and I was expecting the second case to fail just like the first one does, since there is no difference between them in my mind.
>
>
>
> Thanks to all those who replied, but I am afraid you're not touching upon my question. I am

sorry I have not made my point clearly enough.
>
>
>
> 1) In a world where directories can be symbolically linked, it is not necessarily true that './bobo/...' is equal to '.', even when 'bobo' exists. Truncating away 'bobo/..' on the assumption that they are the same is arguably incorrect, but it is certainly confusing if it is done in some cases and not in others!
>
>
>
> 2) In the regular Unix shells, 'ls -ld bobo/..' and 'ls -ld ./bobo/..' both fail, with the error message that the file or directory does not exist.
>
>
>
> 3) FILE_TEST shows the same strange difference between 'bobo/..' and './bobo/..':
>
>
>
> IDL> print, file_test('./bobo/..', /dir)
>
> 1
>
> IDL> print, file_test('bobo/..', /dir)
>
> 0
>
>
>
> It looks like I will have to do what I was hoping to avoid:
>
>
>
> 1) given a path like 'foo/bar/baz/../quux/..', use file_test(x, /dir) on every sub-path x:
>
> file_test('foo', /dir)
>
> file_test('foo/bar', /dir)
>
> file_test('foo/bar/baz', /dir)
>
> file_test('foo/bar/baz/..', /dir)
>
> file_test('foo/bar/baz/../quux', /dir)
>
> file_test('foo/bar/baz/../quux/..', /dir)
>

>
>
> If IDL cannot be relied upon to fail in the next step when one or more of these path components don't exist, then I must test every one of them.
>
>
>
> 2) If all of these tests pass, then
>
>
>
>
> cd 'foo/bar/baz/../quux/..', curr=old
>
> cd, old, curr=pwd
>
>
>
> 3) pwd is now the fully-qualified path to the directory indicated by 'foo/bar/baz/../quux/..'
>
>
>
>
>
>
> Cheers,
>
>
>
> Tom

Well, symbolic links solve some problems, and create some others :-)

Quick and unportable hack: you can get the real path by calling 'realpath' in your Linux C library:

```
IDL> in_path='foo/bar/baz/../quux/..'
IDL> out_path=string(replicate(32b, 1024))
IDL> x=call_external('/lib64/libc-2.11.3.so', 'realpath', in_path, out_path, /ul64_value, value=[1,1], /auto_glue)
```

(adjust it to your system.)

If in_path does not exist, x will be zero, otherwise out_path will contain the real path.

regards,
Lajos