Subject: Any interest in an IDL to Python interface?
Posted by Jason Ferrara on Thu, 28 Feb 2008 16:00:30 GMT
View Forum Message <> Reply to Message

We're thinking of coming out with product that acts as a bridge between IDL and Python, and are trying to get an idea of how much of a demand there is for this sort of thing.

It would make Python modules usable directly from IDL.

Some simple usage examples, meant to show how the interface works, rather than why you might want to use Python from IDL.

Using the Python Imaging Library to load an image, rotate it, and then place it in an IDL array.

```
IDL> numpy=pyimport('numpy')
Loaded DLM: PYTHONFROMIDL.
IDL> pilimage=pyimport('PIL.Image')
IDL> img=pilimage->open('scan.jpeg')
IDL> img=img->rotate(30)
IDL> imgarr=numpy->array(img)
IDL> help, imgarr
IMGARR          BYTE      = Array[3, 850, 864]
IDL> tv, imgarr, /true, order=1
```

Defining and calling an arbitrary python function.

```
IDL> py=pyimport('__main__')
IDL> py->exec, "def mulbytwo(a):"+string(10b)+"    return [x*2 for x
in a]"
IDL> print, py->mulbytwo([1,2,3,4,5])
          2          4          6          8         10
```

Features of the interface:

Python objects (including modules) appear in IDL as IDL objects.

Automatic conversion of method parameters from IDL variables to the appropriate python type.

Automatic conversion of return values to IDL types.

The Python environment runs in the same process as IDL, so parameter passing is fast.

Automatic garbage collection of IDL objects that represent Python objects, so calling OBJ_DESTROY is not required. This makes the objects behave more Python like, so that you can do things like

"img=((pilimage->open('scan.jpeg'))->rotate(30))->convert('L') "
without leaking objects or having to call HEAP_GC.

Would anyone find this useful?

Thanks

Jason Ferrara
Jacquette Consulting, Inc.

Subject: Re: Any interest in an IDL to Python interface?
Posted by Anthony[1] on Mon, 03 Mar 2008 09:19:39 GMT
View Forum Message <> Reply to Message

On Mar 2, 6:57 pm, Michael Aye <kmichael....@googlemail.com> wrote:
> On Mar 1, 3:49 am, metachronist <rkombi...@gmail.com> wrote:
>
>
>
>> On Feb 29, 9:42 am, Jason Ferrara <jason.ferr...@jacquette.com> wrote:
>
>>> On Feb 28, 12:13 pm, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
>
>>>> Hi
>
>>>> it could be nice if something like that is added for all supported OS
>>>> versions not only for windows.
>
>>> Its our plan to support Windows, Linux, and Mac.
>
>>>> python has lots of interesting libraries and is used as scripting
>>>> language in many programs (OpenOffice.org, Blender, Maya, PyMOL, Gimp, etc.)
>
>>>> The problem will be the price or do you have considered to share it free
>>>> of charge?
>
>>> It would be a commercial product. The pricing hasn't been determined
>>> yet.
>
>>>> And another question has to be discussed too. What of idl will be left
>>>> over if such a powerful programming language will be added to idl.
>>>> For example this will give us the possibility to use QT4 for widgets. Or
>>>> we never again do get "the sky is falling down miracle" because of the
>>>> decimal data type. Or we can use pythons standalone webserver or lots of
>>>> math libraries free of charge.
>

>>>> One has to ask himself if an idl program is mostly based on python
>>>> modules why he has to use idl and does not do the whole job in python
>>>> then? There are not much differences between both languages.
>
>>>> How would the development of idl been continued if we as user could
>>>> always use other libs?
>
>>> I don't see this as being an issue.
>
>>> Each language has its own strengths and weaknesses. Its not a one
>>> size
>>> fits all thing. Interoperability between languages means you can mix
>>> and
>>> match as best solves your problem, rather than having to pick one and
>>> then struggle with it for the things its not good with.
>
>>> And more libraries accessible from IDL makes IDL more useful, not
>>> less.
>
>> First of all this is an excellent idea.. But just curious.. what is
>> the memory overhead in computationally intensive apps? Any bench
>> marks? I personally am a big python advocate and learning more
>> everyday! Thanks for any additional info that you can provide.
>> Best wishes,
>> /rk
>
> Just adding:
> Me too, plz. I love Python and would at least like to have some
> interoperability.
> I'm also eagerly waiting for a Python-SPICE interface, then SPICE
> results could be fed directly into IDL's powerful plotting routines.
> Or IDL's results could be fed via Python into this Python-supported 3D
> rendering program Blender.
> Go ahead with it!
> Best regards,
> Michael

Great idea - definitely worth having access to both Python and IDL
functionality at the same time.

For myself, I think I'd generally want to use IDL within Python,
rather than Python within IDL - though it would be good to be able to
do both.

There are a couple of ways of running IDL from Python. E.g.,

pyIDL <http://www.its.caltech.edu/~mmckerns/software.html> - looks
great, but I couldn't get it to compile on my machine.

pIDLy <http://astronomy.sussex.ac.uk/~anthonys/pidly/> or <http://pypi.python.org/pypi/pIDLy/> - a module I wrote as a wrapper on Pexpect. More likely to compile, but pretty slow for large arrays etc., not likely to get much faster, and at an early stage of development.

Cheers,

Anthony

Subject: Re: Any interest in an IDL to Python interface?
Posted by m_schellens on Mon, 03 Mar 2008 14:12:48 GMT
View Forum Message <> Reply to Message

GDL
http://sourceforge.net/projects/gnudatalanguage

has this functionality.
Actually both ways: One can call python from GDL similar as described and GDL can be compiled as a python module.
The drawback is that GDL only supports python numarray yet.

Cheers,
Marc

On Feb 28, 5:00 pm, Jason Ferrara <jason.ferr...@jacquette.com> wrote:
> We're thinking of coming out with product that acts as a bridge
> between IDL and Python, and are trying to get an idea of how much of a
> demand there is for this sort of thing.
>
> It would make Python modules usable directly from IDL.
>
> Some simple usage examples, meant to show how the interface works,
> rather than why you might want to use Python from IDL.
>
> Using the Python Imaging Library to load an image, rotate it, and then
> place it in an IDL array.
> IDL> numpy=pyimport('numpy')
> Loaded DLM: PYTHONFROMIDL.
> IDL> pilimage=pyimport('PIL.Image')
> IDL> img=pilimage->open('scan.jpeg')
> IDL> img=img->rotate(30)
> IDL> imgarr=numpy->array(img)
> IDL> help, imgarr
> IMGARR BYTE = Array[3, 850, 864]

> IDL> tv, imgarr, /true, order=1
>
> Defining and calling an arbitrary python function.
> IDL> py=pyimport('__main__')
> IDL> py->exec, "def mulbytwo(a):"+string(10b)+"   return [x*2 for x
> in a]"
> IDL> print, py->mulbytwo([1,2,3,4,5])
>   2   4   6   8   10
>
> Features of the interface:
>
>   Python objects (including modules) appear in IDL as IDL objects.
>
>   Automatic conversion of method parameters from IDL variables to the
> appropriate python type.
>
>   Automatic conversion of return values to IDL types.
>
>   The Python environment runs in the same process as IDL, so parameter
> passing is fast.
>
>   Automatic garbage collection of IDL objects that represent Python
> objects, so calling OBJ_DESTROY is not required. This makes the
> objects behave more Python like, so that you can do things like
> "img=((pilimage->open('scan.jpeg'))->rotate(30))->convert('L') "
> without leaking objects or having to call HEAP_GC.
>
> Would anyone find this useful?
>
> Thanks
>
> Jason Ferrara
> Jacquette Consulting, Inc.

Subject: Re: Any interest in an IDL to Python interface?
Posted by R.Bauer on Mon, 03 Mar 2008 14:26:00 GMT
View Forum Message <> Reply to Message

m_schellens@hotmail.com schrieb:
> GDL
> http://sourceforge.net/projects/gnudatalanguage
>
> has this functionality.
> Actually both ways: One can call python from GDL similar as described
> and GDL can be compiled as a python module.
> The drawback is that GDL only supports python numarray yet.
>

yeah but because it is completly open source someone who is interested
could help to implement some other modules.

Btw. do you have done a request for gdl as a summer of code project?

cheers
Reimar

>
>
> On Feb 28, 5:00 pm, Jason Ferrara <jason.ferr...@jacquette.com> wrote:
>> We're thinking of coming out with product that acts as a bridge
>> between IDL and Python, and are trying to get an idea of how much of a
>> demand there is for this sort of thing.
>>
>> It would make Python modules usable directly from IDL.
>>
>> Some simple usage examples, meant to show how the interface works,
>> rather than why you might want to use Python from IDL.
>>
>> Using the Python Imaging Library to load an image, rotate it, and then
>> place it in an IDL array.
>> IDL> numpy=pyimport('numpy')
>> Loaded DLM: PYTHONFROMIDL.
>> IDL> pilimage=pyimport('PIL.Image')
>> IDL> img=pilimage->open('scan.jpeg')
>> IDL> img=img->rotate(30)
>> IDL> imgarr=numpy->array(img)
>> IDL> help, imgarr
>> IMGARR   BYTE   = Array[3, 850, 864]
>> IDL> tv, imgarr, /true, order=1
>>
>> Defining and calling an arbitrary python function.
>> IDL> py=pyimport('__main__')
>> IDL> py->exec, "def mulbytwo(a):"+string(10b)+"   return [x*2 for x
>> in a]"
>> IDL> print, py->mulbytwo([1,2,3,4,5])
>>   2   4   6   8   10
>>
>> Features of the interface:
>>
>>   Python objects (including modules) appear in IDL as IDL objects.
>>
>>   Automatic conversion of method parameters from IDL variables to the

>> appropriate python type.
>>
>> Automatic conversion of return values to IDL types.
>>
>> The Python environment runs in the same process as IDL, so parameter
>> passing is fast.
>>
>> Automatic garbage collection of IDL objects that represent Python
>> objects, so calling OBJ_DESTROY is not required. This makes the
>> objects behave more Python like, so that you can do things like
>> "img=((pilimage->open('scan.jpeg'))->rotate(30))->convert('L') "
>> without leaking objects or having to call HEAP_GC.
>>
>> Would anyone find this useful?
>>
>> Thanks
>>
>> Jason Ferrara
>> Jacquette Consulting, Inc.
>

Subject: Re: Any interest in an IDL to Python interface?
Posted by R.Bauer on Mon, 03 Mar 2008 16:36:48 GMT
View Forum Message <> Reply to Message

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

m_schellens@hotmail.com schrieb:
> GDL
> http://sourceforge.net/projects/gnudatalanguage
>
> has this functionality.
> Actually both ways: One can call python from GDL similar as described
> and GDL can be compiled as a python module.
> The drawback is that GDL only supports python numarray yet.
>
> Cheers,
> Marc

btw. it was quite easy to install on my olpc XO
yum install gdl plplot

it seems not such easy on my SuSE box.

cheers
Reimar

>
>
> On Feb 28, 5:00 pm, Jason Ferrara <jason.ferr...@jacquette.com> wrote:
>> We're thinking of coming out with product that acts as a bridge
>> between IDL and Python, and are trying to get an idea of how much of a
>> demand there is for this sort of thing.
>>
>> It would make Python modules usable directly from IDL.
>>
>> Some simple usage examples, meant to show how the interface works,
>> rather than why you might want to use Python from IDL.
>>
>> Using the Python Imaging Library to load an image, rotate it, and then
>> place it in an IDL array.
>> IDL> numpy=pyimport('numpy')
>> Loaded DLM: PYTHONFROMIDL.
>> IDL> pilimage=pyimport('PIL.Image')
>> IDL> img=pilimage->open('scan.jpeg')
>> IDL> img=img->rotate(30)
>> IDL> imgarr=numpy->array(img)
>> IDL> help, imgarr
>> IMGARR   BYTE   = Array[3, 850, 864]
>> IDL> tv, imgarr, /true, order=1
>>
>> Defining and calling an arbitrary python function.
>> IDL> py=pyimport('__main__')
>> IDL> py->exec, "def mulbytwo(a):"+string(10b)+"   return [x*2 for x
>> in a]"
>> IDL> print, py->mulbytwo([1,2,3,4,5])
>>   2   4   6   8   10
>>
>> Features of the interface:
>>
>>   Python objects (including modules) appear in IDL as IDL objects.
>>
>>   Automatic conversion of method parameters from IDL variables to the
>> appropriate python type.
>>
>>   Automatic conversion of return values to IDL types.
>>
>>   The Python environment runs in the same process as IDL, so parameter
>> passing is fast.
>>
>>   Automatic garbage collection of IDL objects that represent Python
>> objects, so calling OBJ_DESTROY is not required. This makes the
>> objects behave more Python like, so that you can do things like

>> "img=((pilimage->open('scan.jpeg'))->rotate(30))->convert('L') "
>> without leaking objects or having to call HEAP_GC.
>>
>> Would anyone find this useful?
>>
>> Thanks
>>
>> Jason Ferrara
>> Jacquette Consulting, Inc.
>

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (GNU/Linux)
Comment: Using GnuPG with SUSE - http://enigmail.mozdev.org

iD8DBQFHzCkf5aOc3Q9hk/kRArmZAJ9GOav/SsAUeukW2gpYhG31ighXSwCe POth
bul1Z3PQzx+usfjdPNXm520=
=AkqP
-----END PGP SIGNATURE-----