Subject: Re: the type parser
Posted by Chris Lee on Sun, 15 May 2005 12:26:00 GMT
View Forum Message <> Reply to Message

In article <d674k0$3dl$1@zam602.zam.kfa-juelich.de>, "Reimar Bauer"
<R.Bauer@fz-juelich.de> wrote:

> Dear all
> I am searching for the function which tells idl what kind of type a
> variable gets by it's assignment. e.g.
> ....
> cheers
> Reimar

Hi Reimar ,

I'm not sure I understand, do you want the type of a constant value? (e.g
1, 2.3, 50000 ). Wouldn't SIZE or SIZE(/type) do this for you?

```
size(1, /type) -> 2          ;INT
size(50000, /type) -> 3   ;LONG
size(1.0, /type) -> 4       ;FLOAT
compile_opt idl2
size(1,/type) -> 3          ;LONG
```

Chris.

Subject: Re: the type parser
Posted by Ken Mankoff on Sun, 15 May 2005 15:19:46 GMT
View Forum Message <> Reply to Message

On Sun, 15 May 2005, Reimar Bauer wrote:
> I am searching for the function which tells idl what kind of type
> a variable gets by it's assignment. e.g.

IDL> print, SIZE(X,/TNAME)
Or the /type keyword works too.

-k.

--
http://spacebit.dyndns.org/

Subject: Re: the type parser

Christopher Lee wrote:

> In article <d674k0$3dl$1@zam602.zam.kfa-juelich.de>, "Reimar Bauer"
> <R.Bauer@fz-juelich.de> wrote:
>
>
>> Dear all
>> I am searching for the function which tells idl what kind of type a
>> variable gets by it's assignment. e.g.
>> ....
>> cheers
>> Reimar
>
> Hi Reimar ,
>
> I'm not sure I understand, do you want the type of a constant value? (e.g
> 1, 2.3, 50000 ). Wouldn't SIZE or SIZE(/type) do this for you?
>
> size(1, /type) -> 2 ;INT
> size(50000, /type) -> 3 ;LONG
> size(1.0, /type) -> 4 ;FLOAT
> compile_opt idl2
> size(1,/type) -> 3 ;LONG
>
> Chris.

Hi Chris,

I look for the opposite direction. If I have a text file e.g.

[call_read]
a=1
b=50000
c=1.0
d='some text'
e=[1,2,3,4]

then I have strings and want to know which type is needed if they would be assigned to variables.

With the read_ini() from our library* I could read this easily into a structure using simple types. At the moment I do determine between float, long or string. But I don't do it the same way as idl it did.
I am intereseted how idl it did and if it is implemented somewhere without the usage of EXECUTE. For example if it is named type:

IDL> print,type('1B')
IDL> 1
IDL> print,type('1')
IDL> 2
IDL> print,type('50000')
IDL> 3
IDL> print,type('1.0')
IDL> 4
IDL> print,type('some text')
IDL> 7
IDL> print,type('[1,2,3,4]')
IDL> 2 2 2 2

One way could be to call type in a different idl session and to use the file
as input like a journal. But I don't like to do this.

*The read_ini()/write_ini will be published with the next version in
probably two months.

cheers
Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 ========== ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: the type parser
Posted by R.Bauer on Sun, 15 May 2005 16:14:32 GMT
View Forum Message <> Reply to Message

The read_ini(), write_ini could be used later on to pass parameters from a
text file into a routine compiled for virtual machine if the type routine
is completed.

cheers
Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/

========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

Subject: Re: the type parser
Posted by marc schellens[1] on Mon, 16 May 2005 09:26:24 GMT
View Forum Message <> Reply to Message

IDL scans the strings.
Its not really difficult but due to the many ways constants can be
defined
in IDL it gets a little bit complicated.
Here is the relevant GDL
(http://sourceforge.net/projects/gnudatalanguage)
code (in antlr, http://www.antlr.org) wich is 100% compatible to IDL.
The assignment to _ttype sets the determined type.
If you translate this into IDL you got what you need.
HDH,
marc

```
protected
D
 : ('0'..'9')
 ;

protected
L
 : ('a'..'z'|'_')
 ;

protected
H
 : ('a'..'f'|'0'..'9')
 ;

protected
O
 : ('0'..'7')
 ;

protected
EXP
 : ('e' ('+'|'-')? (D)+)
 ;

protected
```

```
DBL_E
    : 'd' { $setText( "E");}
    ;

protected
DBL
 : (DBL_E (('+'|'-')? (D)+)?)
 ;

protected
CONSTANT_HEX_BYTE:;
protected
CONSTANT_HEX_LONG:;
protected
CONSTANT_HEX_LONG64:;
protected
CONSTANT_HEX_I:; // integer or larger
protected
CONSTANT_HEX_INT:;
protected
CONSTANT_HEX_ULONG:;
protected
CONSTANT_HEX_ULONG64:;
protected
CONSTANT_HEX_UI:;
protected
CONSTANT_HEX_UINT:;
protected
CONSTANT_BYTE:;
protected
CONSTANT_LONG:;
protected
CONSTANT_LONG64:;
protected
CONSTANT_I:; // integer or larger if necessary
protected
CONSTANT_INT:;
protected
CONSTANT_ULONG:;
protected
CONSTANT_ULONG64:;
protected
CONSTANT_UI:;
protected
CONSTANT_UINT:;
protected
CONSTANT_OCT_BYTE:;
protected
```

```
CONSTANT_OCT_LONG:;
protected
CONSTANT_OCT_LONG64:;
protected
CONSTANT_OCT_I:; // integer or larger if necessary
protected
CONSTANT_OCT_INT:;
protected
CONSTANT_OCT_ULONG:;
protected
CONSTANT_OCT_ULONG64:;
protected
CONSTANT_OCT_UI:;
protected
CONSTANT_OCT_UINT:;
protected
CONSTANT_FLOAT:;
protected
CONSTANT_DOUBLE:;
protected
STRING_LITERAL:;
protected
DOT:;

CONSTANT_OR_STRING_LITERAL
 // could be a string, but octals have priority
 : ('\"'(O)+ ( 'b' | 's' | "us" | "ub" | 'l' | 'u' | "ul" )?) =>
  ('\"'! (O)+  { _ttype=CONSTANT_OCT_I; }  // DEFINT32
   ( 's'!  { _ttype=CONSTANT_OCT_INT; }
   | 'b'!  { _ttype=CONSTANT_OCT_BYTE; }
   | 'u'!      { _ttype=CONSTANT_OCT_UI; }   // DEFINT32
   | "us"!     { _ttype=CONSTANT_OCT_UINT; }
   | "ub"!  { _ttype=CONSTANT_OCT_BYTE; }
   | 'l'!      { _ttype=CONSTANT_OCT_LONG; }
   | "ll"!     { _ttype=CONSTANT_OCT_LONG64; }
   | "ul"!  { _ttype=CONSTANT_OCT_ULONG; }
   | "ull"! { _ttype=CONSTANT_OCT_ULONG64; }
   )?)
 | ('\"(H)+'\" ( 'x' | "xs" | "xb" | "xl" | "xu" | "xus" | "xub" |
"xul")) =>
  ('\"! (H)+ '\"! 'x'!
    (        { _ttype=CONSTANT_HEX_I; } // DEFINT32
   | 's'!  { _ttype=CONSTANT_HEX_INT; }
   | 'b'!  { _ttype=CONSTANT_HEX_BYTE; }
   | 'u'!     { _ttype=CONSTANT_HEX_UI; }   // DEFINT32
   | "us"!     { _ttype=CONSTANT_HEX_UINT; }
                | "ub"!  { _ttype=CONSTANT_HEX_BYTE; }
   | 'l'!     { _ttype=CONSTANT_HEX_LONG; }
```

```
  | "ll"!     { _ttype=CONSTANT_HEX_LONG64; }
  | "ul"!     { _ttype=CONSTANT_HEX_ULONG; }
    | "ull"! { _ttype=CONSTANT_HEX_ULONG64; }
  ))
| ('\''(O)+'\'' ( 'o' | "os" | "ol" | "ou" | "oul")) =>
 ('\''! (O)+ '\''! 'o'!
  (        { _ttype=CONSTANT_OCT_I; } // DEFINT32
  | 's'!      { _ttype=CONSTANT_OCT_INT; }
  | 'b'!      { _ttype=CONSTANT_OCT_BYTE; }
  | 'u'!      { _ttype=CONSTANT_OCT_UI; }   // DEFINT32
  | "us"!     { _ttype=CONSTANT_OCT_UINT; }
  | "ub"!     { _ttype=CONSTANT_OCT_BYTE; }
  | 'l'!      { _ttype=CONSTANT_OCT_LONG; }
  | "ll"!     { _ttype=CONSTANT_OCT_LONG64; }
  | "ul"!     { _ttype=CONSTANT_OCT_ULONG; }
  | "ull"! { _ttype=CONSTANT_OCT_ULONG64; }
  ))
// strings in IDL do not need trailing '' or "
| '\"'! (~('\"'|'\r'|'\n')| '\"' '\"'! )*
 ( '\"'!
 |
 )           { _ttype=STRING_LITERAL; }
| '\''! (~('\''|'\r'|'\n')| '\'' '\''!  )*
 ( '\''!
 |
 )         { _ttype=STRING_LITERAL; }
| (((D)+ (DBL | '.'(D)*(DBL))) | '.'(D)+(DBL)) =>
 (
  (
   (D)+
   ( DBL
   | '.'(D)*(DBL)
   )
  )
 | '.'(D)+(DBL))
     { _ttype=CONSTANT_DOUBLE; }
| (((D)+ (EXP | '.'(D)*(EXP)?)) | '.'(D)+(EXP)?) =>
 (
  (
   (D)+
   ( EXP
   | '.'(D)*(EXP)?
   )
  )
 | '.'(D)+(EXP)?)
 { _ttype=CONSTANT_FLOAT; }
| '.'              { _ttype=DOT; }
| (D)+       { _ttype=CONSTANT_I; }
```

```
  ( 's'!         { _ttype=CONSTANT_INT; }
  | 'b'!   { _ttype=CONSTANT_BYTE; }
  | 'u'('s')?! { _ttype=CONSTANT_UINT; }
  | "ub"!   { _ttype=CONSTANT_BYTE; }
  | 'l'!      { _ttype=CONSTANT_LONG; }
  | "ll"!      { _ttype=CONSTANT_LONG64; }
  | "ul"!      { _ttype=CONSTANT_ULONG; }
  | "ull"!     { _ttype=CONSTANT_ULONG64; }
  )?
 ;
```

---

Subject: Re: the type parser
Posted by R.Bauer on Fri, 20 May 2005 08:16:30 GMT
View Forum Message <> Reply to Message

m_schellens@hotmail.com wrote:
> IDL scans the strings.
> Its not really difficult but due to the many ways constants can be
> defined
> in IDL it gets a little bit complicated.
> Here is the relevant GDL
> (http://sourceforge.net/projects/gnudatalanguage)
> code (in antlr, http://www.antlr.org) wich is 100% compatible to IDL.
> The assignment to _ttype sets the determined type.
> If you translate this into IDL you got what you need.
> HDH,
> marc
>
>

Thanks,

cheers
Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
 ---------- -------
        a IDL library at ForschungsZentrum Juelich
  http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
 ========== =======

---

Subject: Re: the type parser
Posted by R.Bauer on Wed, 15 Jun 2005 09:36:27 GMT
View Forum Message <> Reply to Message

Dear all

just again a question on this

I have recognized that it is easier to do nothing

plot,['1.2','1.3','1.4'],['1.2','1.3','1.4']

I wonder a little bit about this but it seems to go right. By this the
routine who is called will set the type.

```
IDL> a='1.23'
IDL> print,a+0.1
      1.33000

IDL> b='99.99'
IDL> print,b gt 100
   0
```

What else should be checked?

cheers

Reimar

Reimar Bauer wrote:
> m_schellens@hotmail.com wrote:
>
>> IDL scans the strings.
>> Its not really difficult but due to the many ways constants can be
>> defined
>> in IDL it gets a little bit complicated.
>> Here is the relevant GDL
>> (http://sourceforge.net/projects/gnudatalanguage)
>> code (in antlr, http://www.antlr.org) wich is 100% compatible to IDL.
>> The assignment to _ttype sets the determined type.
>> If you translate this into IDL you got what you need.
>> HDH,
>> marc
>>
>>

>
> Thanks,
>
> cheers
> Reimar
>

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
---------- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
========== =======