Subject: Re: Using IDL from a perl script
Posted by Michele Dondi on Fri, 19 Oct 2007 18:33:58 GMT
View Forum Message <> Reply to Message

On Fri, 19 Oct 2007 11:02:41 -0700, James Everton
<james.everton@gmail.com> wrote:

```
> open(IDL, "|/usr/local/bin/idl -32");
> print IDL <<"ENDIDL";
> var1 = '$myPerlVar1'
> var2 = '$myPerlVar2'
> .compile my_web_script
> my_web_script, var1, var2
> ENDIDL
>
```
> I'm wondering if there is any way to pass from this IDL block (say for
> instance that the my_web_script procedure was actually a function and
> was assigned to a new variable returnVar) out to the containing perl
> script. For those that don't know IDL, I'm quite certain that you can

IPC::Open{2,3}

Michele
--
```
{$_=pack'B8'x25,unpack'A8'x32,$a^=sub{pop^pop}->(map substr
(($a||=join'',map--$|x$_,(unpack'w',unpack'u','G^<R<Y]*YB='
.'KYU;*EVH[.FHF2W+#"\Z*5TI/ER<Z`S(G.DZZ9OX0Z')=~/./g)x2,$_,
256),7,249);s/[^\w,]/ /g;$ \=/^J/?$/:"\r";print,redo}#JAPH,
```

Subject: Re: Using IDL from a perl script
Posted by A. Sinan Unur on Fri, 19 Oct 2007 18:37:29 GMT
View Forum Message <> Reply to Message

James Everton <james.everton@gmail.com> wrote in
news:1192816961.304312.109840@q3g2000prf.googlegroups.com:

> I'm developing some web scripts in perl that interact with a database
> through various pre-written IDL procedures here at my work. The
> problem I'm having is getting arguments passed between the two.
> Passing from the perl into the IDL code is easy enough because I'm
> using the open( ) procedure and simply writing the strings as a block:
>
```
>  open(IDL, "|/usr/local/bin/idl -32");
>  print IDL <<"ENDIDL";
>  var1 = '$myPerlVar1'
>  var2 = '$myPerlVar2'
```

> .compile my_web_script
> my_web_script, var1, var2
> ENDIDL

Instead of typing the code yourself, you are having perl type the code.
This is not what I understand from "passing arguments".

> I'm wondering if there is any way to pass from this IDL block (say for
> instance that the my_web_script procedure was actually a function and
> was assigned to a new variable returnVar) out to the containing perl
> script.

There is a serious disconnect in your thinking here. You just opened a
write-only pipe to the /usr/local/bin/idl process. Even if you could
pass some return values (by having the IDL script throw an error), it
would be cumbersome.

Maybe perldoc perlipc can help but you need to improve your
understanding of how interprocess communications work. Why are you
writing the script above? Why wouldn't you just type your IDL program in
an editor rather than generating these scripts using Perl?

Sinan

--
A. Sinan Unur <1usa@llenroc.ude.invalid>
(remove .invalid and reverse each component for email address)
clpmisc guidelines: <URL:http://www.augustmail.com/~tadmc/clpmisc.shtml>

Subject: Re: Using IDL from a perl script
Posted by jkj on Sat, 20 Oct 2007 23:16:08 GMT

View Forum Message <> Reply to Message

On Oct 19, 1:02 pm, James Everton <james.ever...@gmail.com> wrote:
> Hi everybody,
>
> I'm developing some web scripts in perl that interact with a database
> through various pre-written IDL procedures here at my work. The
> problem I'm having is getting arguments passed between the two.
> Passing from the perl into the IDL code is easy enough because I'm
> using the open( ) procedure and simply writing the strings as a block:
>

I've done this a bunch and found it to be very useful... here's a bit
more of what I find works:

```
use IPC::Open2;

$progName = "idl";
open2(READIDL, WRITEIDL, $progName) or die "Could not begin \"$progName
\"\n";

# Then you can cause IDL to execute commands like this:
print WRITEIDL "value = 13\n";

# I find that output coming back from IDL is cluttered, so before
#   asking IDL to return a value I ask it to spit out "junk":
print WRITEIDL "print, \"junk\"\n";
while(!(<READIDL> =~ /junk/)){
}

# Now I can get the value I want back asking IDL to output it:
print WRITEIDL "print, value\n";
$value = <READIDL>;
chomp($value);
```

...hope that helps,
-Kevin

Subject: Re: Using IDL from a perl script
Posted by James Everton on Mon, 22 Oct 2007 20:05:05 GMT

View Forum Message <> Reply to Message

```
On Oct 20, 5:16 pm, jkj <ke...@vexona.com> wrote:
> On Oct 19, 1:02 pm, James Everton <james.ever...@gmail.com> wrote:
>
>> Hi everybody,
>
>> I'm developing some web scripts in perl that interact with a database
>> through various pre-written IDL procedures here at my work.  The
>> problem I'm having is getting arguments passed between the two.
>> Passing from the perl into the IDL code is easy enough because I'm
>> using the open( ) procedure and simply writing the strings as a block:
>
> I've done this a bunch and found it to be very useful... here's a bit
> more of what I find works:
>
> use IPC::Open2;
>
> $progName = "idl";
> open2(READIDL, WRITEIDL, $progName) or die "Could not begin \"$progName
> \"\n";
>
```

```
>  # Then you can cause IDL to execute commands like this:
>  print WRITEIDL "value = 13\n";
>
>  # I find that output coming back from IDL is cluttered, so before
>  #   asking IDL to return a value I ask it to spit out "junk":
>  print WRITEIDL "print, \"junk\"\n";
>  while(!(<READIDL> =~ /junk/)){
>
>  }
>
>  # Now I can get the value I want back asking IDL to output it:
>  print WRITEIDL "print, value\n";
>  $value = <READIDL>;
>  chomp($value);
>
>  ...hope that helps,
>  -Kevin
```

Thanks so much! Took me a while to get a feel for how the handles
were being set up, but it worked like a charm!
As for the IDL junk, I found that setting my $progName to 'idl &> /dev/
null' got rid of all the junk as well as compilation print outs.

Pipes are your friend :-)
- James