Subject: Re: Who called that procedure?
Posted by Michael Galloy on Tue, 03 Feb 2009 17:23:56 GMT
View Forum Message <> Reply to Message

wlandsman wrote:
> This is a minor problem but it has cost me a half-hour a couple of
> times.
>
> I run RESOLVE_ALL to make sure that I am including all necessary
> procedures when distributing software.   One procedure may yield ~200
> compiled supporting procedures, and I see one -- say obsolete.pro --
> that should not be being used any more.   So I want to know which of
> the 200 procedures is still calling obsolete.pro.   There does not
> seem to be any pattern to the order of procedures displayed by
> RESOLVE_ALL so that does not help.   What would be nicest I suppose
> would be a tree diagram of all the dependencies.
>
> In the end, I can simply grep the 200 procedures to see which one has
> a call to obsolete.pro. But is there a better way?   Thanks, -- Wayne

I would like to add the creation of such dependency trees to IDLdoc at
some point. In particular, they would be handy when giving the minimal
amount of source to someone. Concerns about method calls,
call_procedure/call_function/execute, and distinguishing a function call
from an array indexing have always made me delay implementing it.

I usually end up just searching (although I have replaced grep with ack,
it conveniently ignores .svn directories and some other niceties to
reduce the false positives).

Another strategy is to put a "compile_opt obsolete" statement in
MY_OBSOLETE_ROUTINE and then set:

```
IDL> !warn.obs_routines = 1
```

Now, calls to MY_OBSOLETE_ROUTINE will generate syntax errors when
compiled (identifying their exact location in the error message).

Mike
--
www.michaelgalloy.com
Tech-X Corporation
Associate Research Scientist

Subject: Re: Who called that procedure?
Posted by R.Bauer on Wed, 04 Feb 2009 08:50:28 GMT

mgalloy schrieb:
> wlandsman wrote:
>> This is a minor problem but it has cost me a half-hour a couple of
>> times.
>>
>> I run RESOLVE_ALL to make sure that I am including all necessary
>> procedures when distributing software. One procedure may yield ~200
>> compiled supporting procedures, and I see one -- say obsolete.pro --
>> that should not be being used any more. So I want to know which of
>> the 200 procedures is still calling obsolete.pro. There does not
>> seem to be any pattern to the order of procedures displayed by
>> RESOLVE_ALL so that does not help. What would be nicest I suppose
>> would be a tree diagram of all the dependencies.
>>
>> In the end, I can simply grep the 200 procedures to see which one has
>> a call to obsolete.pro. But is there a better way? Thanks, -- Wayne
>
> I would like to add the creation of such dependency trees to IDLdoc at
> some point. In particular, they would be handy when giving the minimal
> amount of source to someone. Concerns about method calls,
> call_procedure/call_function/execute, and distinguishing a function call
> from an array indexing have always made me delay implementing it.
>
> I usually end up just searching (although I have replaced grep with ack,
> it conveniently ignores .svn directories and some other niceties to
> reduce the false positives).
>
> Another strategy is to put a "compile_opt obsolete" statement in
> MY_OBSOLETE_ROUTINE and then set:
>
> IDL> !warn.obs_routines = 1
>
> Now, calls to MY_OBSOLETE_ROUTINE will generate syntax errors when
> compiled (identifying their exact location in the error message).
>
> Mike

Is it something like this?

http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl
_html/dbase/download/a_and_b_called_from.html

http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _html/idl_work_libraries.htm

(look at the symbols with the arrow down)

May be I can extract the code then.

cheers

Reimar

Subject: Re: Who called that procedure?
Posted by Michael Galloy on Wed, 04 Feb 2009 15:31:43 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:
> mgalloy schrieb:
>> wlandsman wrote:
>>> This is a minor problem but it has cost me a half-hour a couple of
>>> times.
>>>
>>> I run RESOLVE_ALL to make sure that I am including all necessary
>>> procedures when distributing software. One procedure may yield ~200
>>> compiled supporting procedures, and I see one -- say obsolete.pro --
>>> that should not be being used any more.   So I want to know which of
>>> the 200 procedures is still calling obsolete.pro.   There does not
>>> seem to be any pattern to the order of procedures displayed by
>>> RESOLVE_ALL so that does not help. What would be nicest I suppose
>>> would be a tree diagram of all the dependencies.
>>>
>>> In the end, I can simply grep the 200 procedures to see which one has
>>> a call to obsolete.pro. But is there a better way? Thanks, -- Wayne
>> I would like to add the creation of such dependency trees to IDLdoc at
>> some point. In particular, they would be handy when giving the minimal
>> amount of source to someone. Concerns about method calls,
>> call_procedure/call_function/execute, and distinguishing a function call
>> from an array indexing have always made me delay implementing it.
>>
>> I usually end up just searching (although I have replaced grep with ack,
>> it conveniently ignores .svn directories and some other niceties to
>> reduce the false positives).
>>
>> Another strategy is to put a "compile_opt obsolete" statement in
>> MY_OBSOLETE_ROUTINE and then set:
>>
>> IDL> !warn.obs_routines = 1
>>
>> Now, calls to MY_OBSOLETE_ROUTINE will generate syntax errors when
>> compiled (identifying their exact location in the error message).
>>
>> Mike
>
> Is it something like this?

>
> http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _html/dbase/download/a_and_b_called_from.html
>
> http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _html/idl_work_libraries.htm
>
> (look at the symbols with the arrow down)
>
> May be I can extract the code then.
>
> cheers
>
> Reimar

Yes, that looks like what I would want (although I would probably list
the routines called by the routine). I assume it handles
procedures/functions called in the normal way (no methods and no
routines called with CALL_PROCEDURE, EXECUTE, etc.)? Just those would
still be quite useful.

--
www.michaelgalloy.com
Tech-X Corporation
Associate Research Scientist

Subject: Re: Who called that procedure?
Posted by R.Bauer on Thu, 05 Feb 2009 09:37:49 GMT
View Forum Message <> Reply to Message

mgalloy schrieb:
> Reimar Bauer wrote:
>> mgalloy schrieb:
>>> wlandsman wrote:
>>>> This is a minor problem but it has cost me a half-hour a couple of
>>>> times.
>>>>
>>>> I run RESOLVE_ALL to make sure that I am including all necessary
>>>> procedures when distributing software. One procedure may yield ~200
>>>> compiled supporting procedures, and I see one -- say obsolete.pro --
>>>> that should not be being used any more. So I want to know which of
>>>> the 200 procedures is still calling obsolete.pro. There does not
>>>> seem to be any pattern to the order of procedures displayed by
>>>> RESOLVE_ALL so that does not help. What would be nicest I suppose
>>>> would be a tree diagram of all the dependencies.
>>>>
>>>> In the end, I can simply grep the 200 procedures to see which one has
>>>> a call to obsolete.pro. But is there a better way? Thanks, -- Wayne

>>> I would like to add the creation of such dependency trees to IDLdoc at
>>> some point. In particular, they would be handy when giving the minimal
>>> amount of source to someone. Concerns about method calls,
>>> call_procedure/call_function/execute, and distinguishing a function call
>>> from an array indexing have always made me delay implementing it.
>>>
>>> I usually end up just searching (although I have replaced grep with ack,
>>> it conveniently ignores .svn directories and some other niceties to
>>> reduce the false positives).
>>>
>>> Another strategy is to put a "compile_opt obsolete" statement in
>>> MY_OBSOLETE_ROUTINE and then set:
>>>
>>> IDL> !warn.obs_routines = 1
>>>
>>> Now, calls to MY_OBSOLETE_ROUTINE will generate syntax errors when
>>> compiled (identifying their exact location in the error message).
>>>
>>> Mike
>>
>> Is it something like this?
>>
>> http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl
_html/dbase/download/a_and_b_called_from.html
>>
>>
>> http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _html/idl_work_libraries.htm
>>
>>
>> (look at the symbols with the arrow down)
>>
>> May be I can extract the code then.
>>
>> cheers
>>
>> Reimar
>
> Yes, that looks like what I would want (although I would probably list
> the routines called by the routine).

see,
http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _html/dbase/a_and_b_dbase.pro.html

it is the database icon on the idl_work_libraries page. Additional you
can get a sav file or a tgz file by that function also shown on the example.

I assume it handles
> procedures/functions called in the normal way (no methods and no

> routines called with CALL_PROCEDURE, EXECUTE, etc.)? Just those would
> still be quite useful.
>

no it founds everything because uses an idl child process to get that data.

The routine name is catalog_db it is also in that lib.

cheers
Reimar
