Subject: Re: Nice ways to compile
Posted by Ken Mankoff on Sun, 10 Apr 2005 15:45:28 GMT
View Forum Message <> Reply to Message

On Sun, 10 Apr 2005, Robert Barnett wrote:
> I'm looking for an easier way to compile my code into a .sav file.
Me too. I use the same process (roughly). The method seems to work, but it also seems fragile.

> + Complile the package and depenancies to a save file and run in
> an IDL VM "sandbox" environment.

> I expect to be able to compile my IDL packages with a single UNIX
> script (step marked as a +). I also expect that my compilation
> depends solely on the contents of the filesystem rather than the
> current state of the IDL command prompt. This ensures that I can
> always return to a snapshot of my idl directory as it was when I
> compiled my code.

I think it is tricky to get it to rely on the contents of the filesystem. There are known issues when it will not track down dependencies that you or I would see reading the code (is this what you mean by filesystem?). Hence your request for a compiler directive.

> For example, I currently compile like the following:
I use IDL with other software that I build from emacs/CLI by calling make on a standard *nix Makefile. So 1 command: "make". At some point in the dependency chain I call

```
  idl make.pro
```

which is

```
; makefile for EVA
; produces eva.sav
.reset_session
.com my_map_set
.com eva
.com eva_earth__define
.com eva_ocean__define
.com eva_data__define
.com eva_image__define
;;; this part grows as my code library grows.
RESOLVE_ALL
spawn, 'echo $HOSTNAME', h
if h = 'random' then save, /routines, EMBEDDED=..., file='eva.sav' else $
  save, /routines, file='eva.sav' ; no license
```

$cp eva.sav ../IDL/bin/
exit
; is there any way to determine if I am a real interactive IDL
; session or a unix CLI "$ idl foo.pro" version? Because
; this exit should be conditional :)

> Perhaps I am looking at this problem completely wrong. I would
> expect there to be some way I can append a compiler directive
> which indicates that blah__define.pro requires
> 'idlgrlegend__define'. Compiler directives are not really in the
> IDL idiom, so I'm wondering what other choices I might have.
They are in the idiom. See ?COMPILE_OPT
But none does what you want that I know of.

I'm thinking there is some way (with a shell script and grep?) to solve your dependencies automagically, until IDL does it correctly itself.

-k.

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

Subject: Re: Nice ways to compile
Posted by Robert Barnett on Mon, 11 Apr 2005 00:41:36 GMT
View Forum Message <> Reply to Message

Thanks Ken,

I'm so glad that I'm not alone. I think that the best thing is to compile as best I can. I'll take the philosophy that "she'll be right mate" unless I hear otherwise.

I've had thoughts of categorizing the code in the IDL directory using soft links. This would allow my to inlcude categories of code in my sav file.

Something like

ln -s /usr/local/rsi/idl_6.0/lib/idlgr*define* ~/idl/IDLgrObjects/

might suffice.

--

nrb@ Robbie Barnett
imag Research Assistant
wsahs Nuclear Medicine & Ultrasound
nsw Westmead Hospital
gov Sydney Australia
au +61 2 9845 7223

Subject: Re: Nice ways to compile
Posted by mmiller3 on Mon, 11 Apr 2005 18:42:43 GMT
View Forum Message <> Reply to Message

I make save files by creating an idl script that compiles all the codes that my application uses (or at least those that I can remember!). I include resolve_all and resolve_all, /class for all the classes that I need. Then I do a save and exit. So making a save set is just a command like "idl make_save_file.pro" (see below). This has all the elegance of makefile that is maintained by hand, which is to say, very little. I've considered trying to make a preprocessor that creates header files so I can use makedepend. If incremental compilation were possible (that is, loading compiled code, instead of having to compile in order to make code available), that would be useful, but with IDL, it doesn't seem neccessary.

Usually I create my make_save_file.pro's from listings of all the *.pro files in the directories where I know I've got code components for a given application. That makes save files with cruft that is never used, but it hasn't (yet) left me with anything missing.

Mike

```
;; make_save_file.pro
.compile first_file
.compile second_file
.compile third_file
;; ...more here as needed...
.compile last_file

.compile main

resolve_all
resolve_all, class=['med_imageobj', $
                    'registered_med_imageobj',$
```

```
                'indypet_kinetics', $
                'ipvis_kinetics', $
                'showprogress' $
               ]

save, /routines, filename='main.sav'

exit
```

Subject: Re: Nice ways to compile
Posted by JD Smith on Tue, 12 Apr 2005 01:18:58 GMT
View Forum Message <> Reply to Message

On Mon, 11 Apr 2005 13:42:43 -0500, Michael A. Miller wrote:

> I make save files by creating an idl script that compiles all the
> codes that my application uses (or at least those that I can
> remember!). I include resolve_all and resolve_all, /class for
> all the classes that I need. Then I do a save and exit. So
> making a save set is just a command like "idl make_save_file.pro"
> (see below). This has all the elegance of makefile that is
> maintained by hand, which is to say, very little. I've
> considered trying to make a preprocessor that creates header
> files so I can use makedepend. If incremental compilation were
> possible (that is, loading compiled code, instead of having to
> compile in order to make code available), that would be useful,
> but with IDL, it doesn't seem neccessary.
>
> Usually I create my make_save_file.pro's from listings of all the
> *.pro files in the directories where I know I've got code
> components for a given application. That makes save files with
> cruft that is never used, but it hasn't (yet) left me with
> anything missing.

When creating a SAV file, I go through and compile all my code one procedure at a time, as below. First I regularize the path, then I go through all ".pro" files, pruning any rejects, and compile them. Only then do I call RESOLVE_ALL, and write out the binary. Since path caching tends to mess up dynamic path changes, I tend to do this in a new IDL session.

Does anyone else get TRNLOG and SETLOG errors when RESOLVE_ALL runs? As near as I can tell, these are some obsolete VMS routines that seem to sneak in somehow (through NasaLIB, I believe).

JD

```
pro compile_cubism
  FORWARD_FUNCTION TRNLOG
  @cubism_dir
  ps=path_sep()
  bindir=filepath(ROOT=cubism_dir,'bin')
  if file_test(filepath(ROOT=bindir,'cubism_vm.sav')) then $
     file_delete,filepath(ROOT=bindir,'cubism_vm.sav')

  ;; Go one level up and compile everything
  sourcepath=cubism_dir
   sourcepath=strmid(sourcepath,0,strpos(sourcepath,ps,/REVERSE _SEARCH))

  ;; Normalize path
  path=strsplit(!PATH,':',/EXTRACT)
  wh=where((sp=strpos(path,ps+'nasa',/REVERSE_SEARCH)) ne -1)
  nasa=strmid(path[wh[0]],0,sp[wh[0]]+5)
  !PATH=expand_path('<IDL_DEFAULT>'+":+"+sourcepath+':+'+nasa)

  files=file_search(sourcepath,'*.pro')
  skip_files=['cubism_dir','cubism_version','compile_cubism', $
              ps+'scraps'+ps,'CVS'+ps]
  resolve_routine,'XManager',/COMPILE_FULL_FILE
  skip=0
  for i=0,n_elements(files)-1 do begin
     for j=0,n_elements(skip_files)-1 do begin
        if strpos(files[i],skip_files[j]) ne -1 then begin
           print,'Skipping '+files[i]
           skip=1
           break
        endif
     endfor
     if skip then begin
        skip=0
        continue
     endif
     print,'Compiling '+files[i]
     routine=strmid(files[i],0,strpos(files[i],".pro",/REVERSE_SEARCH))
      routine=strmid(routine,strpos(files[i],ps,/REVERSE_SEARCH)+1 )
      resolve_routine,routine,/EITHER,/NO_RECOMPILE,/COMPILE_FULL_ FILE
  endfor
  resolve_all,/CONTINUE_ON_ERROR
   save,/ROUTINES,FILENAME=filepath(ROOT=bindir,'cubism_vm.sav' )
end
```

Subject: Re: Nice ways to compile
Posted by Jeff Guerber on Tue, 12 Apr 2005 18:59:47 GMT

On Mon, 11 Apr 2005, JD Smith wrote:

> Does anyone else get TRNLOG and SETLOG errors when
> RESOLVE_ALL runs? As near as I can tell, these are some obsolete VMS
> routines that seem to sneak in somehow (through NasaLIB, I believe).

Yes, also DELLOG. I forget exactly where they were, but there seemed to still be references to them within IDL itself. I created small dummy procedures:

```
pro setlog
end
```

etc., just so they'd have definitions.

(I can't really add anything on the real issue. We have a simple script that makes a list of all the .pro files in the directory and writes an IDL procedure with a .compile for each, which is a solution someone else has already mentioned.)

Jeff Guerber

Subject: Re: Nice ways to compile
Posted by biophys on Sat, 16 Apr 2005 09:35:25 GMT

I use skip_routines and it works fine:

```
resolve_all, class=['MYVMCONSOLE', 'FSC_PSCONFIG'], $
   skip_routines=['TRNLOG','DELLOG','SETLOG']
```

Jeff Guerber wrote:
> On Mon, 11 Apr 2005, JD Smith wrote:
>
>> Does anyone else get TRNLOG and SETLOG errors when
>> RESOLVE_ALL runs? As near as I can tell, these are some obsolete VMS
>> routines that seem to sneak in somehow (through NasaLIB, I believe).
>
> Yes, also DELLOG. I forget exactly where they were, but there seemed
> to still be references to them within IDL itself. I created small

dummy
> procedures:
>
> pro setlog
> end
>
> etc., just so they'd have definitions.
>
> (I can't really add anything on the real issue. We have a simple
> script that makes a list of all the .pro files in the directory and
writes
> an IDL procedure with a .compile for each, which is a solution
someone
> else has already mentioned.)
>
> Jeff Guerber
