Subject: Re: Config files
Posted by marc schellens[1] on Thu, 09 Feb 2006 11:08:33 GMT
View Forum Message <> Reply to Message

Probably you are better off using
SAVE and RESTORE.
It is not in an ascii file then,
but I could only think of the need if you must
change the values from outside IDL.

Marc

Subject: Re: Config files
Posted by Ben Panter on Thu, 09 Feb 2006 12:15:53 GMT
View Forum Message <> Reply to Message

m_schellens@hotmail.com wrote:
> Probably you are better off using
> SAVE and RESTORE.
> It is not in an ascii file then,
> but I could only think of the need if you must
> change the values from outside IDL.

I would use save and restore, but start out with a batch file which can
do the save and restore for you. Something like this:

set_config.pro:

```
;change these parts:
input = myinputdata.dat
output = myoutputwillgohere.dat
aconstant = 1.0
anarray = [1,2,3]

;leave this alone:
save, input, output, aconstant, anarray, filename='settings.sav'
```

When you start a run, call @set_config and then later on in the code do
a restore, 'settings.sav'. I do something similar, but save the settings
in a struct rather than loose parameters. That way it's easier to pass
around to any routines that need it later on, and I'm less likely to use
the parameter names for something else by mistake.

Hope that's helpful,

Ben

--
Ben Panter, Garching, Germany.
Email false, http://www.benpanter.co.uk
or you could try ben at ^^^^^^^^^^^^^^^

Subject: Re: Config files
Posted by Edd Edmondson on Thu, 09 Feb 2006 12:18:02 GMT
View Forum Message <> Reply to Message

Ben Panter <me@privacy.net> wrote:
> I would use save and restore, but start out with a batch file which can
> do the save and restore for you. Something like this:

> set_config.pro:

> ;change these parts:
> input = myinputdata.dat
> output = myoutputwillgohere.dat
> aconstant = 1.0
> anarray = [1,2,3]

> ;leave this alone:
> save, input, output, aconstant, anarray, filename='settings.sav'

> When you start a run, call @set_config and then later on in the code do
> a restore, 'settings.sav'. I do something similar, but save the settings
> in a struct rather than loose parameters. That way it's easier to pass
> around to any routines that need it later on, and I'm less likely to use
> the parameter names for something else by mistake.

> Hope that's helpful,

Cracking idea, and I'm kicking myself for not having thought of it!

--
Edd

Subject: Re: Config files
Posted by savoie on Thu, 09 Feb 2006 15:31:20 GMT
View Forum Message <> Reply to Message

Edd <eddedmondson@hotmail.com> writes:

> Ben Panter <me@privacy.net> wrote:
>> I would use save and restore, but start out with a batch file which can
>> do the save and restore for you. Something like this:
>
>> set_config.pro:
>
>> ;change these parts:
>> input = myinputdata.dat
>> output = myoutputwillgohere.dat
>> aconstant = 1.0
>> anarray = [1,2,3]
>
>> ;leave this alone:
>> save, input, output, aconstant, anarray, filename='settings.sav'
>
>
>> When you start a run, call @set_config and then later on in the code do
>> a restore, 'settings.sav'. I do something similar, but save the settings
>> in a struct rather than loose parameters. That way it's easier to pass
>> around to any routines that need it later on, and I'm less likely to use
>> the parameter names for something else by mistake.
>
>> Hope that's helpful,
>
> Cracking idea, and I'm kicking myself for not having thought of it!

I'm doing the same thing. That's a great idea.

I wrote a subclass of Craig Markwardt's HASHTABLE__define.pro, mhs_iniread__define, that lets me pass a configuation file into a routine and get back an object with pretty normal functionality.

```
ini = Obj_New('mhs_iniread', file='inifile.txt')

value = ini->get('value')
```

The only problem is that it's not that robust. I know the quirks, and I work around them rather than fixing my code.

It's just a thought.

Matt

--
Matthew Savoie - Scientific Programmer
National Snow and Ice Data Center
(303) 735-0785 http://nsidc.org

Subject: Re: Config files
Posted by news.qwest.net on Thu, 09 Feb 2006 16:53:31 GMT
View Forum Message <> Reply to Message

"Edd" <eddedmondson@hotmail.com> wrote in message
news:dsf79ifam1@news.ox.ac.uk...
> I'd like to shift my rather clunky code with too many hardcoded
> constants over to using some sort of configuration file, ideally
> something in a sensible ASCII format. I'd want to pass my code the
> filename, and then it could read the file and pair things up in a
> structure, so I might have a file like
>
> input = myinputdata.dat
> output = myoutputwillgohere.dat
> aconstant = 1.0
> anarray = [1,2,3]
>
> and read it with something like
> foo=readconfig('configdata.txt')
> and get a datastructure from it like
> foo.input='myinputdata.dat'
> foo.aconstant=1.0 etc...
>
> Shirley I don't need to reinvent the wheel? Anything functionally
> vaguely similar would be very handy.
>
> --
> Edd

A while ago I tried to make a singleton config object.
Couldn't really do it, but what I ended up doing was making
a system variable of an object (of an array of structures),
and had a initilization routine that checked to see if it already existed.

```
pro initialize_qscatinfo,verbose=verbose

defsysv,'!qscatinfo',exist=exist

If exist then begin
 if keyword_set(verbose) then print,'variable already defined: '
endif else begin
 if keyword_set(verbose) then print,'defining variable: '
 defsysv,'!qscatinfo',OBJ_NEW('qscat_info')
endelse

end ; procedure
```

The object was an array of structures, and it had the usual "get" method

that was based on keywords:
return = !qscatinfo->get(/keyword)

Cheers,
bob

Subject: Re: Config files
Posted by mmiller3 on Thu, 09 Feb 2006 19:05:11 GMT
View Forum Message <> Reply to Message

>>>> > "savoie" == savoie <savoie@nsidc.org> writes:

> ini = Obj_New('mhs_iniread', file='inifile.txt')

Have you considered using APP_USER_DIR to handle where you put
the configuration file? I'm about to add some configuration
handling to an application and am thinking of using it to make
sure each user can easily save their own parameters.

As for parsing the configuration data, I've considered using save
files, but it seems to me that the simplest format would be to
have the configuration saved as IDL code. Then it can be edited
by hand if the user wants or reloaded by EXECUTE'ing the contents
of the file.

Mike

Subject: Re: Config files
Posted by Robert Barnett on Fri, 10 Feb 2006 04:43:58 GMT
View Forum Message <> Reply to Message

Another way is to do it with structures. Just keep your settings in a
structure variable which is defined in a common block. I have included
some example code to read and write structures to/from a text file.
Mind the line feeds inserted by Google.

```
pro rt_textstructRead, struct, filename
	print, "Reading", filename
	if (N_ELEMENTS(struct) EQ 0) then return
	if (~ file_test(filename, /READ)) then message, "Cannot read
text struct
ure file " + string(filename)
	openr,lun1,filename,/GET_LUN
	names = tag_names(struct)
```

```
        pair = ''
        while (not EOF(lun1)) do begin
                readf, lun1, pair
                hashi = strpos(pair,'#')
                if (hashi GE 0) then pair = strmid(pair,0,hashi)
                eqi = strpos(pair,'=')
                if (eqi GE 1) then begin
                        name = strupcase(strtrim(strmid(pair,0,eqi),2))
                        for i=0,N_TAGS(struct)-1 DO BEGIN
                                if (names[i] EQ name) then begin
                                        value = struct.(i)
                                        setvalue =
strtrim(strmid(pair,eqi+1,str
len(pair)-eqi),2)
                                        if (setvalue ne '') then begin
                                           reads, setvalue, value
                                           struct.(i) = value
                                        endif
                                endif
                        endfor
                endif
        endwhile
        free_lun, lun1
end


pro rt_textstructWrite, struct, filename
        if (N_ELEMENTS(struct) EQ 0) then return
        openw,lun1,filename,/GET_LUN
        names = tag_names(struct)
        for i=0,N_TAGS(struct)-1 DO BEGIN
                printf, lun1, names[i], "       = ",struct.(i)
        endfor
        printf, lun1
        free_lun, lun1
end
```

Robbie
