Subject: IDL/FORTRAN File Reading
Posted by robparker23 on Thu, 04 Jun 2009 17:04:22 GMT
View Forum Message <> Reply to Message

This might be a bit of a strange request but I'm hoping we have some multi-skilled people here who might be able to help.

I have a file written by some fortran code as a "f77_unformatted" file.

I've managed to figure out how to read this in in IDL and I've done as follows:

```
header = strarr(80,1)
openr,unit,filename, /f77_unf,/get_lun

i=0
while ~ eof(unit) do begin

readu,unit,header
b = 0.0d0
c = 0.0d0
e = 0.0d0
f = 0L
g = 0.0
readu,unit,b,c,e,f,nlo

if f EQ -99 then break

a=dblarr(f,1)
readu,unit,a


case i of
0: BEGIN
data=a
END
data=[data, a]
END
endcase

i=1
endwhile

close, unit
free_lun, unit
```

This reads in b,c,d,e,f and then f is used to determine how big a is

and then that chunk of data is read in. It then repeats with a new
value of f being read in which defines a new chunk of a and so on
until the EOF.

That's probably a lot simpler than I described it.

Anyway my problem is that whilst I can do this in IDL, ironically I
can't figure out how to do it in FORTRAN. As i'm just about at the
"hello world" stage that's not surprising but I thought it would
simply be a case of defining my variables as the correct type and then
just using the fortran READ command but that spits out the wrong
values. I was hoping that someone capable in both IDL and FORTRAN
might be able to "translate" between the two for me.

---

## Subject: Re: IDL/FORTRAN File Reading
## Posted by penteado on Tue, 09 Jun 2009 02:47:06 GMT

View Forum Message <> Reply to Message

On Jun 4, 2:04 pm, robparke...@googlemail.com wrote:

```
> This might be a bit of a strange request but I'm hoping we have some
> multi-skilled people here who might be able to help.
>
> I have a file written by some fortran code as a "f77_unformatted"
> file.
>
> I've managed to figure out how to read this in in IDL and I've done as
> follows:
>
> header = strarr(80,1)
> openr,unit,filename, /f77_unf,/get_lun
>
> i=0
> while ~ eof(unit) do begin
>
> readu,unit,header
> b = 0.0d0
> c = 0.0d0
> e = 0.0d0
> f = 0L
> g = 0.0
> readu,unit,b,c,e,f,nlo
>
> if f EQ -99 then break
>
> a=dblarr(f,1)
> readu,unit,a
>
```

>  case i of
>  0: BEGIN
>  data=a
>  END
>  data=[data, a]
>  END
>  endcase
>
>  i=1
>  endwhile
>
>  close, unit
>  free_lun, unit
>
>  This reads in b,c,d,e,f and then f is used to determine how big a is
>  and then that chunk of data is read in. It then repeats with a new
>  value of f being read in which defines a new chunk of a and so on
>  until the EOF.
>
>  That's probably a lot simpler than I described it.
>
>  Anyway my problem is that whilst I can do this in IDL, ironically I
>  can't figure out how to do it in FORTRAN. As i'm just about at the
>  "hello world" stage that's not surprising but I thought it would
>  simply be a case of defining my variables as the correct type and then
>  just using the fortran READ command but that spits out the wrong
>  values. I was hoping that someone capable in both IDL and FORTRAN
>  might be able to "translate" between the two for me.

Your description seems to be sufficient, but it would be easier if
there was a sample of the kind of file to read.

---

Subject: Re: IDL/FORTRAN File Reading
Posted by penteado on Tue, 09 Jun 2009 18:30:31 GMT
View Forum Message <> Reply to Message

On Jun 8, 11:47 pm, pp.pente...@gmail.com wrote:
>  On Jun 4, 2:04 pm, robparke...@googlemail.com wrote:
>
>
>
>>  This might be a bit of a strange request but I'm hoping we have some
>>  multi-skilled people here who might be able to help.
>
>>  I have a file written by some fortran code as a "f77_unformatted"
>>  file.
>

```
>> I've managed to figure out how to read this in in IDL and I've done as
>> follows:
>
>> header = strarr(80,1)
>> openr,unit,filename, /f77_unf,/get_lun
>
>> i=0
>> while ~ eof(unit) do begin
>
>> readu,unit,header
>> b = 0.0d0
>> c = 0.0d0
>> e = 0.0d0
>> f = 0L
>> g = 0.0
>> readu,unit,b,c,e,f,nlo
>
>> if f EQ -99 then break
>
>> a=dblarr(f,1)
>> readu,unit,a
>
>> case i of
>> 0: BEGIN
>> data=a
>> END
>> data=[data, a]
>> END
>> endcase
>
>> i=1
>> endwhile
>
>> close, unit
>> free_lun, unit
>
>> This reads in b,c,d,e,f and then f is used to determine how big a is
>> and then that chunk of data is read in. It then repeats with a new
>> value of f being read in which defines a new chunk of a and so on
>> until the EOF.
>
>> That's probably a lot simpler than I described it.
>
>> Anyway my problem is that whilst I can do this in IDL, ironically I
>> can't figure out how to do it in FORTRAN. As i'm just about at the
>> "hello world" stage that's not surprising but I thought it would
>> simply be a case of defining my variables as the correct type and then
>> just using the fortran READ command but that spits out the wrong
```

>> values. I was hoping that someone capable in both IDL and FORTRAN
>> might be able to "translate" between the two for me.
>
> Your description seems to be sufficient, but it would be easier if
> there was a sample of the kind of file to read.

This is untested, since I did not have a sample file to test it with,
so there may be some details wrong. But something along those lines
should do it:

```
subroutine readuf77(data,filename)
implicit none
double precision, intent(inout), allocatable :: data(:) !where the
read values will be
character(*), intent(in) :: filename
!dummy variables:
double precision :: b,c,e
integer :: f,nlo
real, allocatable :: a(:),tmp(:) !where the values will be read into
!control variables
integer :: un,i,ios,cnt,nhd
logical :: op

!constants
nhd=80 !number of header lines to skip

!find the first available unit and open the file to it
un=7
do
  inquire(unit=un,opened=op)
  if (.not. op) exit !if unit is not open, it should be fine to use
  un=un+1
enddo
 open(unit=un,file=filename,action='read',form='unformatted')

if (allocated(data)) deallocate(data)
!read the values
ios=0
cnt=0
do while (ios==0)
  do i=1,nhd; read(unit=un,iostat=ios); enddo !skip the nhd header
lines
  read(unit=un,iostat=ios)b,c,e,f,nlo !read the number of elements (f)
to be read into a
  if (f==-99) exit
  allocate(a(f))
  read(unit=un,iostat=ios)a !read the f values into a
  if (cnt==0) then !allocate data for the first time
    allocate(data(f))
    data=a
  else
    allocate(tmp(cnt)) !place to keep a copy of data's contents it is
reallocated
    tmp=data
    deallocate(data)
    allocate(data(cnt+f))
    data(1:cnt)=tmp
    deallocate(tmp)
    data(cnt+1:cnt+f)=a
    cnt=cnt+f !update the count of elements read
  endif
  deallocate(a)
enddo

close(unit=un)

end subroutine readuf77
```