Subject: Lots of files
Posted by lasse on Fri, 16 Mar 2007 17:36:01 GMT
View Forum Message <> Reply to Message

Hi there,

I have a file containing magnetic field data from 31 stations in the
following form

stat01 lots of data
stat02 lots of data
.
.
.
stat31 lots of data
stat01 lots of data
stat02 lots of data
.
.
.
i.e. each line contains data for one station.

I would like to split the one file into 31. In fact, I would like to
open all 32 files, loop through the big file and put the data
according to the index into the small file. However, IDL only lets me
open 28 files at a time, right?
IDL Help for Get_lun says: The file unit number obtained is in the
range 100 to 128.
So I end up opening and closing the according small file during each
loop which works great, however it is excruciatingly slow due to all
the waiting for the hard disk.

Ah, and the number of stations varies, sometimes its 21, sometimes 29,
most of the time 30, I never know what it is going to be. Therefore, I
can only use the point_lun procedure to skip from data to data
belonging to one station if I parse the stations first. Which I could
do, but maybe one of you has a better idea? Any thoughts?

Cheers
Lasse

Subject: Re: Lots of files
Posted by lasse on Mon, 19 Mar 2007 11:32:18 GMT
View Forum Message <> Reply to Message

On 18 Mar, 03:59, David Fanning <n...@dfanning.com> wrote:
> Lasse Clausen writes:

```
>>  Well thanks, that works, however it did not bring the speed boost I
>>  had hoped for. So I had another thought: Actually, all data is one
>>  line, not in one line per station as I said earlier. But I know that
>>  each data set is 1440 characters long, so here is the outline of my
>>  code, after I opened all the files:
>
>>  info = file_info(input_filename)
>>  lines = info.size/1440L
>
>>  for i=0L, lines-1L do begin
>>      point_lun, fin, i*1440L
>>      readf, fin, line, format='(A1440)'
>>      ; extracting station name
>>      hstat = strlowcase(strmid(line, 12, 3))
>>      ; find correct file unit
>>      tmp = where(stats eq hstat)
>>      printf, tmp[0]+1, line
>>  endfor
>
>  Well lots of string processing and WHERE's going
>  on here, which I think is what is slowing things
>  down. How about something like this:
>
>   theLines = Assoc(lun, BytArr(1440))
>   maxYear = Max(stats)
>   for I=0L, lines-1L do begin
>       aLine = theLines[I]
>       ; extracting station name
>       hstat = String(aLine[12:15])
>       ; find correct file unit
>       printf, (maxYear-hstat)+1, String(aLine)
>   endfor
>
>  Cheers,
>
>  David
>  --
>  David Fanning, Ph.D.
>  Fanning Software Consulting, Inc.
>  Coyote's Guide to IDL Programming:http://www.dfanning.com/
>  Sepore ma de ni thui. ("Perhaps thou speakest truth.")
```

Yeeeha! This solution (thanks Lajos/David)

```
openr, fin, filename, /get_lun
theLines = Assoc(fin, BytArr(1440))
for i=0L, lines-1L do begin
   aLine = theLines[I]
   hstat = String(aLine[12:14])
```

```
    tmp = where(stats eq hstat)
    printf, tmp[0]+1, string(aline)
endfor
```

goes like sh** off a shovel! I will use that. However, I still do not
know why

```
openr, fin, filename, /get_lun
for i=0L, lines-1L do begin
    point_lun, fin, i*1440L
    readf, fin, line, format='(A1440)'
endwhile
```

(note that all the string processing and where is taken out) takes
about 30 seconds. Well, I supsect the format code, there isn't much
else to supsect. Can anybody explain to me why the above solution is
so slow?

Cheers
Lasse

Subject: Re: Lots of files
Posted by David Fanning on Mon, 19 Mar 2007 14:27:10 GMT
View Forum Message <> Reply to Message

Lasse Clausen writes:

```
>  However, I still do not
>  know why
>
>
>  openr, fin, filename, /get_lun
>  for i=0L, lines-1L do begin
>      point_lun, fin, i*1440L
>      readf, fin, line, format='(A1440)'
>  endwhile
>
>  (note that all the string processing and where is taken out) takes
>  about 30 seconds. Well, I supsect the format code, there isn't much
>  else to supsect. Can anybody explain to me why the above solution is
>  so slow?
```

I'm guessing here, something I never do in a newsgroup
post. ;-)

I'd guess the biggest problem is the READF, rather

than a READU. I think if you took out the POINT_LUN
and made the READF into a READU this would turn out
to be reasonably fast, too, even with a FORMAT keyword.

I think when you do a READF, you *always* read to the
end of the "line", which in this case is the end of the
file. Then, you have to re-position the file pointer for
the next read, etc. I think that's why it is so slow.
There is a lot of churning going on. You could test this
by checking the file pointer position at the end of the
READF. (Use POINT_LUN, -fin, position & Print, position.)
I'd be curious to know. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Lots of files
Posted by David Fanning on Mon, 19 Mar 2007 14:31:55 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Use POINT_LUN, -fin, position & Print, position.

Whoops! Should be:

```
   POINT_LUN, fin, -position & Print, position
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Lots of files

Posted by lasse on Mon, 19 Mar 2007 16:41:58 GMT
View Forum Message <> Reply to Message

On 19 Mar, 14:31, David Fanning <n...@dfanning.com> wrote:
> David Fanning writes:
>> Use POINT_LUN, -fin, position & Print, position.
>
> Whoops! Should be:
>
> POINT_LUN, fin, -position & Print, position

no, first solution was correct, using negative file units reads
pointer location rather than setting it.

aaaanyway...

As David suspected (to be honest, I had that feeling, too), the readf
command reads from a file unit until it reaches a newline, then
formats the string according to the format code given (IDL help says:
On input, read data from the file and format it according to the
format code.). Therefore my code was slow, because it read 4MB of data
and then chucked away ... well, lots of it.

If one uses readu instead, one cannot use format code ("The READU
procedure reads unformatted binary data") but it just reads as many
bytes as fit in the variable provided. Hence

```
line = bytarr(1440)
openr, fin, filename, /get_lun
while not(eof(fin)) do begin
   readu, fin, line
   hstat = String(line[12:14])
   tmp = where(stats eq hstat)
   printf, tmp[0]+1, string(line)
endwhile
```

works like a charm, too. and please note, finally I am able to use the
not(eof(fin)) condition.

Oh, and by the way, the above program runs in 0.05 seconds, whereas
the readf with format takes 75 seconds but that doesn't surprise
anybody, I guess.

Right, that sorted, let's get some work done...
Cheers
Lasse

Subject: Re: Lots of files
Posted by Foldy Lajos on Mon, 19 Mar 2007 17:01:31 GMT
View Forum Message <> Reply to Message

On Mon, 19 Mar 2007, Lasse Clausen wrote:

```
> On 19 Mar, 14:31, David Fanning <n...@dfanning.com> wrote:
>> David Fanning writes:
>>> Use POINT_LUN, -fin, position & Print, position.
>>
>> Whoops! Should be:
>>
>>    POINT_LUN, fin, -position & Print, position
>
> no, first solution was correct, using negative file units reads
> pointer location rather than setting it.
>
> aaaanyway...
>
> As David suspected (to be honest, I had that feeling, too), the readf
> command reads from a file unit until it reaches a newline, then
> formats the string according to the format code given (IDL help says:
> On input, read data from the file and format it according to the
> format code.). Therefore my code was slow, because it read 4MB of data
> and then chucked away ... well, lots of it.
>
> If one uses readu instead, one cannot use format code ("The READU
> procedure reads unformatted binary data") but it just reads as many
> bytes as fit in the variable provided. Hence
>
> line = bytarr(1440)
> openr, fin, filename, /get_lun
> while not(eof(fin)) do begin
>    readu, fin, line
>    hstat = String(line[12:14])
>    tmp = where(stats eq hstat)
>    printf, tmp[0]+1, string(line)
> endwhile
>
> works like a charm, too. and please note, finally I am able to use the
> not(eof(fin)) condition.
>
> Oh, and by the way, the above program runs in 0.05 seconds, whereas
> the readf with format takes 75 seconds but that doesn't surprise
> anybody, I guess.
>
> Right, that sorted, let's get some work done...
> Cheers
> Lasse
```

>
>

Just two little comments:

1. always use ~ (logical negation) instead of NOT (bitwise negation) for eof(). Here is an excerpt from the manual:

Using the NOT Operator

Due to the bitwise nature of the NOT operator, logical negation operations should always use ~ in preference to NOT, reserving NOT exclusively for bitwise computations. Consider a statement such as:

```
   IF ((NOT EOF(lun)) && device_ready) THEN statement
```

which wants to execute statement if the file specified by the variable lun has data remaining, and the variable device_ready is non-zero. When EOF returns the value 1, the expression NOT EOF(lun) yields -2, due to the bitwise nature of the NOT operator. The && operator interprets the value -2 as true, and will therefore attempt to execute statement incorrectly in many cases. The proper way to write the above statement is:

```
   IF ((~ EOF(lun)) && device_ready) THEN statement
```

2. Formatted I/O is line-oriented, as you have discovered. So your 'printf, tmp[0]+1, string(line)' command actually writes 1441 characters (adds a new line at the end). Use writeu instead to keep the original structure (single line file).

regards,
lajos