Subject: Re: slow file-handling
Posted by brian.jackel on Wed, 04 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

> In article <322ABD58.3F93@ic.ac.uk>, Jorn Helbert <jorn@ic.ac.uk> writes:
> |> line by line it takes nearly ten minutes on a sparc twenty. my read
> |> block is the following
> |> b_temp = double([0.,0.,0.])
> |> t_temp = [0.0]
> |> WHILE NOT eof(u1) DO BEGIN
> |> readf,u1,t_temp,foo,foo,foo,b_temp
> |> t_u = [t_u,t_temp]
> |> b_u = [b_u,b_temp]
> |> END

Using FSTAT() to determine the number of lines works just fine on
a PC, your mileage may vary on other platforms. For example

```
IDL>  OPENR,1,file1
IDL> help,/str,fstat(1)
** Structure FSTAT, 12 tags, length=36:
   UNIT            LONG                 1
   NAME            STRING    'f:\midas\aug23.96\millstone\file12.pwr'
   OPEN            BYTE         1
   ISATTY          BYTE         0
   ISAGUI          BYTE         0
   INTERACTIVE     BYTE         0
   READ            BYTE         1
   WRITE           BYTE         0
   TRANSFER_COUNT  LONG                 0
   CUR_PTR         LONG                 0
   SIZE            LONG           2704156
   REC_LEN         LONG                 0
```

Which gives the filesize, and the fact we're at the beginning.
Then make something to read in each line, in my case it's
just an array, you may need a structure of some kind.

```
IDL>  line= FLTARR(710)
IDL>    READF,1,line
```

After that, the filestatus shows:

```
IDL> help,/str,fstat(1)
** Structure FSTAT, 12 tags, length=36:
   UNIT            LONG                 1
   NAME            STRING    'f:\midas\aug23.96\millstone\file12.pwr'
   OPEN            BYTE         1
```

```
ISATTY          BYTE        0
ISAGUI          BYTE        0
INTERACTIVE     BYTE        0
READ            BYTE        1
WRITE           BYTE        0
TRANSFER_COUNT  LONG          710
CUR_PTR         LONG        11362
SIZE            LONG      2704156
REC_LEN         LONG            0
```

Where according to the help file:

TRANSFER_COUNT is
The number of scalar IDL data items transferred in the last input/output operation on the unit. This is set by the following IDL routines: READU, WRITEU, PRINT, PRINTF, READ, and READF. TRANSFER_COUNT is useful when attempting to recover from input/output errors.

and CUR_PTR tells you how many bytes were read in total. So long as there's no extra stuff at the end of the file, just divide REC_LEN by CUR_PTR to get the total number of lines. After that, allocate a large enough block, and read in everything at once.

Please let me know if this doesn't work, I'd been assuming I could move this to other platforms, and would like to know if I'm in for trouble :)

Brian Jackel
