Subject: Read Total lines in an ASCII file
Posted by msmimb on Fri, 13 Dec 2002 17:42:10 GMT
View Forum Message <> Reply to Message

I bet my question seems simple to all of you but...does anybody know
how to read the total number of lines in an ASCII file?

Also, is there any command in IDL such as if (variable_char is char)
then give me a boolean (true or false)?

Thanks a lot!
Maria.

Subject: Re: Read Total lines in an ASCII file
Posted by mchinand on Sat, 14 Dec 2002 02:29:07 GMT
View Forum Message <> Reply to Message

In article <3dfa626d@news.nwl.ac.uk>, <wmc@bas.ac.uk> wrote:
>
> spawn,'cat filename|wc -l',number_of_lines
>
> should work.
>
> -W.
>

or just,

spawn, 'wc -l <filename', number_of_lines

--Mike

--
Michael Chinander
m-chinander@uchicago.edu
Department of Radiology
University of Chicago

Subject: Re: Read Total lines in an ASCII file
Posted by James Kuyper on Sat, 14 Dec 2002 16:57:46 GMT
View Forum Message <> Reply to Message

wmc@bas.ac.uk wrote:
...

> If you are lucky enough to be running under unix, then
>
> spawn,'cat filename|wc -l',number_of_lines

spawn 'wc -l filename', number_of_lines

would seem like a simpler way to do it.

Subject: Re: Read Total lines in an ASCII file
Posted by wmconnolley on Sun, 15 Dec 2002 15:16:09 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@saicmodis.com> wrote:
> wmc@bas.ac.uk wrote:
>> If you are lucky enough to be running under unix, then
>>
>> spawn,'cat filename|wc -l',number_of_lines

> spawn 'wc -l filename', number_of_lines

> would seem like a simpler way to do it.

It would, until you try it ;-) when you'll discover that your
version appends the filename to the line count (if this is *nix
dependent, I retract my sarcasm).

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: Read Total lines in an ASCII file
Posted by Karsten Rodenacker on Mon, 16 Dec 2002 07:39:21 GMT
View Forum Message <> Reply to Message

Maria wrote:

> I bet my question seems simple to all of you but...does anybody know
> how to read the total number of lines in an ASCII file?
>
> Also, is there any command in IDL such as if (variable_char is char)
> then give me a boolean (true or false)?
>

> Thanks a lot!
> Maria.
I am using this structure :
```
  mt = {version:1., $
        datastart:0L, $
        delimiter:0b, $
        missingvalue:!values.f_nan, $
        commentsymbol:'', $
        fieldcount:[1l], $
        fieldtypes:7, $
        fieldnames:'field1', $
        fieldlocations:0l, $
       fieldgroups:[0]}
  aa = read_ascii(filnam, template = mt)
```
to read ascii data with the idl reading routine. May be that helps.
Regards
--

Karsten Rodenacker ()
---------- :-)
GSF - Forschungszentrum Institute of Biomathematics and Biometry
D-85758 Oberschleissheim Postfach 11 29
Tel: +49 89 31873401 | FAX: ...3369 | rodena@gsf.de | Karsten@Rodenacker.de
http://www.gsf.de/ibb/homepages/rodenacker

Subject: Re: Read Total lines in an ASCII file
Posted by rmoss4 on Mon, 16 Dec 2002 16:35:42 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
> *snip*
>
> ... IDL arrays *look* like they can be extended,
> but in fact every time you extend an IDL array you create a new one.
>

What about this then?

array = [TEMPORARY(array), new_element]

Pointless?

Robert Moss, PhD
rmoss4@houston.rr.com

Subject: Re: Read Total lines in an ASCII file

Posted by David Fanning on Mon, 16 Dec 2002 16:48:21 GMT
View Forum Message <> Reply to Message

Robert Moss (rmoss4@houston.rr.com) writes:

> What about this then?
>
> array = [TEMPORARY(array), new_element]
>
> Pointless?

I don't know how memory is managed in IDL except
in the most general terms, but I suspect that
done thousands of times, the syntax above will
inevitably lead to memory fragmentation. I think
getting memory in "chunks", as JD and others recommend,
must be more memory efficient in the long run.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Read Total lines in an ASCII file
Posted by Paul Van Delst[1] on Mon, 16 Dec 2002 17:06:02 GMT
View Forum Message <> Reply to Message

Robert Moss wrote:
>
> Mark Hadfield wrote:
>> *snip*
>>
>> ... IDL arrays *look* like they can be extended,
>> but in fact every time you extend an IDL array you create a new one.
>>
>
> What about this then?
>
> array = [TEMPORARY(array), new_element]
>
> Pointless?

Slow. Especially if your "array" gets biggish. For small files it's a quick and easy method and I use it. When I know I'll be reading large numbers of data, I use the Knight routine DDREAD. Works well.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7274
Fax:(301)763-8545

Subject: Re: Read Total lines in an ASCII file
Posted by Mark Hadfield on Mon, 16 Dec 2002 20:00:42 GMT
View Forum Message <> Reply to Message

"Robert Moss" <rmoss4@houston.rr.com> wrote in message news:yhnL9.168422$Gc.5459794@twister.austin.rr.com...
> Mark Hadfield wrote:

>> *snip*
>>
>> ... IDL arrays *look* like they can be extended,
>> but in fact every time you extend an IDL array you
>> create a new one.
>>
>
> What about this then?
>
> array = [TEMPORARY(array), new_element]

Before this statement executes you have an array occupying n*m bytes of memory, where n is number of elements and m is number of bytes to store one element. After it executes you have an array occupying (n-1)*m bytes. There is no way the new array will fit into the space vacated by the old array, so a new space has to be allocated and all the old elements copied over. This takes significant time (when n is large) and doing it repeatedly tends to leave unusable holes all over the memory space.

There are several things IDL could do to make the above more efficient:

- Allow arrays to be stored non-contiguously (as Numeric Python does, for example). Of course this would open up another can of worms. When your DLM wants to access a non-contiguous array, how

does it no where to find the data?

- Extend arrays in chunks, eg the new array created above would be larger than the old one by a specified increment or a specified ratio (1.5 is a good compromise). Then the array could be extended several more times before the need arose to copy the whole thing. IDL would have to keep track of the fact that not all of the allocated space was currently being used.

- Uh, I am sure there are several more but I can't think of any right now.

I knew this had been discussed on the group some time in the past, so I searched Google. I found a thread that touches on this entitled "How to use pointers instead of arrays" and dated 9-10 Dec 2002!

Those who do not remember history are condemned to repeat it, about once a week.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Read Total lines in an ASCII file
Posted by David Fanning on Mon, 16 Dec 2002 20:24:38 GMT
View Forum Message <> Reply to Message

Mark Hadfield (m.hadfield@niwa.co.nz) writes:

> I knew this had been discussed on the group some time in the past, so I
> searched Google. I found a thread that touches on this entitled "How
> to use pointers instead of arrays" and dated 9-10 Dec 2002!
>
> Those who do not remember history are condemned to repeat it, about
> once a week.

And even writing the answer down doesn't work anymore,
because you keep quoting from articles I didn't know
I had written. :-(

More interesting still, I had a private e-mail
on the VERY same topic (the mis-use of pointers to
do this kind of thing) two days after the newsgroup
article. I've been wondering what strange force is
loose in the Universe this week, when I suddenly

remembered Lord of the Rings, Part II opens Wednesday.
(We already have tickets thanks to a friend of a friend
of my middle son, who works at the theater.)
I'm thinking now that Saruman's army has something to
do with it!

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Read Total lines in an ASCII file
Posted by Mark Hadfield on Mon, 16 Dec 2002 23:06:19 GMT
View Forum Message <> Reply to Message

First, I am pleased to announce that my Motley library is back
on-line. Don't break out the methode traditionelle yet, because its
current home is only temporary, but I do have permission from the
people who own the code (and pay me my salary) to publish the
code. See:

ftp://ftp.niwa.co.nz/incoming/m.hadfield/idl/README.html

Now about this "read total lines in an ASCII file" thing, I have been
inspired to implement 4 methods in my MGH_TXT_RESTORE function, which
reads the contents of a text file into a string array. The methods
are:

0 - Read lines one at a time, accumulating them in a string
array. Extend the array as necessary in increments of 70% or so (see
code). Trim the result before returning.

1 - Read the file once to count the lines, create a result array of
the required size, then read the file again.

2 - Create a result array of size MAX_LINES, read the file, then
trim the result. This method may return fewer lines than the other
methods, as all empty lines are trimmed from the end.

3 - As 0, but accumulate the results in an MGH_Vector object then
move them into a string array at the end.

4 - As 0, but accumulate the results in a string array, extending
the array by just one element each time.

These methods can be exercised by routine
MGH_EXAMPLE_TXT_RESTORE. Here are results for a 20,000 line
uncompressed file on my machine:

0 - 0.38 s
1 - 0.25 s
2 - 0.14 s
3 - 0.82 s
4 - several minutes so far @ 100% CPU utilisation.

Method 2 (the one posted by Med Bennett earlier in this thread) is the speed winner, but I don't like the fact that you need to specify the maximum file size in advance. Method 1, the two-pass method, is surprisingly quick here, but suffers if the file is on a slow network drive. Method 0 is my favourite. Method 3 uses the MGH_Vector object, which uses the same array-growing approach as method 0; I wrote the MGH_Vector object to hide all the ugly array-growing details but in hindsight the performance penalty is too large (and the array-growing code not all that ugly, anyway). Method 4 was included to make the point that it's horrendously slow when the number of array-extending operations exceeds 1000 or so; however I can't be bothered waiting to see how long it will take.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Read Total lines in an ASCII file
Posted by Mark Hadfield on Mon, 16 Dec 2002 23:20:48 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
news:atlme8$mh4$1@newsreader.mailgate.org...
> [Reading a 20,000 line uncompressed text file]
> 0 - 0.38 s
> 1 - 0.25 s
> 2 - 0.14 s
> 3 - 0.82 s
> 4 - several minutes so far @ 100% CPU utilisation.

#4 took 518 s!

--

Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Read Total lines in an ASCII file
Posted by wmconnolley on Tue, 17 Dec 2002 21:40:53 GMT
View Forum Message <> Reply to Message

Mark Hadfield <m.hadfield@niwa.co.nz> wrote:
> Method 1, the two-pass method, is
> surprisingly quick here, but suffers if the file is on a slow network
> drive.

If the read and re-read are close in time (as they should be) the file
will still be in whatever cache the o/s uses, and the second read (as
far as the o/s is concerned) will be much faster. On the ex-dec alphas
I use, faster by a factor of 10 than the first. Whether putting IDL in
between harms this gain I don't know. This may well survive being on a
network drive (I'm assuming what you say above is a guess, rather than
experience? Forgive me if not)

ps: did you try method 5, ie spawn wc -l, then use method 0?

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: Read Total lines in an ASCII file
Posted by Rick Towler on Tue, 17 Dec 2002 22:32:27 GMT
View Forum Message <> Reply to Message

<wmc@bas.ac.uk> wrote
> Mark Hadfield <m.hadfield@niwa.co.nz> wrote:
>> Method 1, the two-pass method, is
>> surprisingly quick here, but suffers if the file is on a slow network
>> drive.
>
> If the read and re-read are close in time (as they should be) the file
> will still be in whatever cache the o/s uses, and the second read (as
> far as the o/s is concerned) will be much faster. On the ex-dec alphas
> I use, faster by a factor of 10 than the first. Whether putting IDL in
> between harms this gain I don't know. This may well survive being on a

> network drive (I'm assuming what you say above is a guess, rather than
> experience? Forgive me if not)

You are assuming that whatever file you are reading will fit into the cache which wouldn't always be the case. To complicate things you have the hard disk drive cache, possibly a cache in the controller, and a cache at the OS level. Each hardware/OS combination will yield different results. So it would be safe to say that YMWV with method 1.

Even if Mark was guessing about the performance of method 1 over the network it is a very safe assumption. Network latency, bandwidth limitations, and server load will always impact read/write performance. Again, the extent depends largely on the hardware/OS/network combination but since most of us aren't lucky enough to have a rack of 15,000 rpm ultra-160 SCSI drives running in RAID 0+1 networked via myrianet we generally have to wait a little longer for files from the server than files stored locally ;)

-Rick

Subject: Re: Read Total lines in an ASCII file
Posted by Mark Hadfield on Tue, 17 Dec 2002 23:22:19 GMT
View Forum Message <> Reply to Message

<wmc@bas.ac.uk> wrote in message news:3dff99e5@news.nwl.ac.uk...
> Mark Hadfield <m.hadfield@niwa.co.nz> wrote:
>> Method 1, the two-pass method, is
>> surprisingly quick here, but suffers if the file is on a slow network
>> drive.
>
> If the read and re-read are close in time (as they should be) the file
> will still be in whatever cache the o/s uses, and the second read (as
> far as the o/s is concerned) will be much faster.

Yes. Actually, my test penalises the two-pass method relative to all the other one-pass methods, because the file is created just before it is read, so is already in cache.

> This may well survive being on a network drive

Depends on the network protocol and parameters. My network connection doesn't seem to do read caching very well.

> ps: did you try method 5, ie spawn wc -l, then use method 0?

I think you mean my method 1 (ie two passes: read the file once to count the lines, create the result array, then read the file again to get the data into the array).

So I revisited method 1, comparing three ways of counting the lines in the file:

1a - Count lines with IDL readf statements in a while loop
1b - Count lines by spawning "wc -l"
1c - Count lines with IDL 5.6 FILE_LINES function

and here are the times taken to read the same 20,000-line, uncompressed file on my hard drive

1a 0.24 s
1b 0.32 s
1c 0.09 s

FILE_LINES is the clear winner. (Isn't it a pity it doesn't accept a COMPRESS keyword!)

Spawning "wc -l" is the slowest. Note that this is on Windows 2000 with the Cygwin "wc" command. Unix is much faster at spawning subprocesses than Windows, so method 1b may be competitive there.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Read Total lines in an ASCII file
Posted by Paul Woodford on Wed, 18 Dec 2002 03:40:39 GMT
View Forum Message <> Reply to Message

Would it be possible to find the length of the file, read it into to a byte array, and then convert it to text?

Paul, who is too lazy to figure it out himself

Subject: Re: Read Total lines in an ASCII file
Posted by Mark Hadfield on Wed, 18 Dec 2002 21:04:18 GMT
View Forum Message <> Reply to Message

"Paul Woodford" <cpwoodford@spamcop.net> wrote in message news:cpwoodford-C4345E.22403817122002@corp.supernews.com...
> Would it be possible to find the length of the file, read it into to
> a byte array, and then convert it to text?

Yes, but:

- If you take the 1D byte array that would result from reading the file and convert it to a string, then you don't get a string array, you just get a string with line-separator characters in it. So there's a bit of splitting to be done, and you really should handle the various line separators supported by the different platforms.

- It doesn't work on compressed files, because you don't know how many bytes there are in a compressed file until you've read it. So you have to read the byte data in chunks, trap the error when the final read hits the end of the file, and join the chunks together.

> Paul, who is too lazy to figure it out himself

Mark, who has thought about it, but can't be bothered actually trying it.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Read Total lines in an ASCII file
Posted by wmconnolley on Wed, 18 Dec 2002 22:12:34 GMT
View Forum Message <> Reply to Message

Mark Hadfield <m.hadfield@niwa.co.nz> wrote:
> I think you mean my method 1 (ie two passes: read the file once to count the
> lines, create the result array, then read the file again to get the data
> into the array).

Oops yes.

> So I revisited method 1, comparing three ways of counting the lines in the
> file:

> 1a - Count lines with IDL readf statements in a while loop
> 1b - Count lines by spawning "wc -l"
> 1c - Count lines with IDL 5.6 FILE_LINES function

> and here are the times taken to read the same 20,000-line, uncompressed file
> on my hard drive

> 1a 0.24 s
> 1b 0.32 s

> 1c 0.09 s

> FILE_LINES is the clear winner. (Isn't it a pity it doesn't accept a
> COMPRESS keyword!)

> Spawning "wc -l" is the slowest. Note that this is on Windows 2000 with the
> Cygwin "wc" command. Unix is much faster at spawning subprocesses than
> Windows, so method 1b may be competitive there.

How interesting. Thanks for doing all this. I would certainly hope that
spawn would be a bit faster under unix. I have a slight feeling that there
are options to spawn that can make it a bit faster (something about not
copying the env perhaps?).

But also intereting to see that wc -l is not quite as o/s specific as I'd
assumed!

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: Read Total lines in an ASCII file
Posted by thompson on Wed, 18 Dec 2002 22:51:57 GMT
View Forum Message <> Reply to Message

wmc@bas.ac.uk writes:

 (stuff deleted)

>> Spawning "wc -l" is the slowest. Note that this is on Windows 2000 with the
>> Cygwin "wc" command. Unix is much faster at spawning subprocesses than
>> Windows, so method 1b may be competitive there.

> How interesting. Thanks for doing all this. I would certainly hope that
> spawn would be a bit faster under unix. I have a slight feeling that there
> are options to spawn that can make it a bit faster (something about not
> copying the env perhaps?).

The way this is done in Unix is to use the /noshell command, e.g.

```
 spawn, 'wc -l '+filename, result  ;Slower
 spawn, ['wc','-l',filename], result, /noshell ;Faster
```

Bill Thompson

Subject: Re: Read Total lines in an ASCII file
Posted by Paul Sorenson on Fri, 20 Dec 2002 04:30:54 GMT
View Forum Message <> Reply to Message

result = string(read_binary("myfile.txt"))

-Paul Sorenson

"Paul Woodford" <cpwoodford@spamcop.net> wrote in message
news:cpwoodford-C4345E.22403817122002@corp.supernews.com...
> Would it be possible to find the length of the file, read it into to a
> byte array, and then convert it to text?
>
> Paul, who is too lazy to figure it out himself

----------== Posted via Newsfeed.Com - Uncensored Usenet News ==----------
http://www.newsfeed.com The #1 Newsgroup Service in the World!
-----= Over 100,000 Newsgroups - Unlimited Fast Downloads - 19 Servers =-----

Subject: Re: Read Total lines in an ASCII file
Posted by condor on Sat, 21 Dec 2002 00:14:51 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
news:<atqt2j$58j$1@newsreader.mailgate.org>...
> "Paul Woodford" <cpwoodford@spamcop.net> wrote in message
> news:cpwoodford-C4345E.22403817122002@corp.supernews.com...
>> Would it be possible to find the length of the file, read it into to
>> a byte array, and then convert it to text?
>
> Yes, but:
>
> - If you take the 1D byte array that would result from reading the
> file and convert it to a string, then you don't get a string array,
> you just get a string with line-separator characters in it. So
> there's a bit of splitting to be done, and you really should handle
> the various line separators supported by the different platforms.

As far as I recall, the OP just wanted to know the number of lines,
not necessarily try to convert them into anything. The only deviation
from the usual 10b linefeed out there on idl'ish platforms is the DOS
[10b,13b] LF/CR, right? Or do VMS systems do yet something different?
How do the various suggested methods hold up on VMS?

If the LF and CR/LF are the only two, the only thing you'd have to do
is counting the number of 10b in the byte-filed:

```
 f = read_binary('Big_honking_example_file')
 h = histogram(f)
 print,h[10]
   1479054
```

If you're really intent on accessing the individual data items in the
file, you could retain the reverse indices of the histogram for a
handy field of pointers to each individual line that can be converted
into a string at will...

> - It doesn't work on compressed files, because you don't know how
> many bytes there are in a compressed file until you've read it. So
> you have to read the byte data in chunks, trap the error when the
> final read hits the end of the file, and join the chunks together.
>
>> Paul, who is too lazy to figure it out himself
>
> Mark, who has thought about it, but can't be bothered actually trying
> it.

Subject: Re: Read Total lines in an ASCII file
Posted by Jeff Guerber on Sat, 21 Dec 2002 02:41:17 GMT
View Forum Message <> Reply to Message

On 20 Dec 2002, Big Bird wrote:

> As far as I recall, the OP just wanted to know the number of lines,
> not necessarily try to convert them into anything. The only deviation
> from the usual 10b linefeed out there on idl'ish platforms is the DOS
> [10b,13b] LF/CR, right? Or do VMS systems do yet something different?
> How do the various suggested methods hold up on VMS?

I haven't used VMS in a number of years (thank goodness!), but IIRC it
had about 5 or 6 different file types, most of which could be used for
ASCII files... Ahh, here's my old copy of "Programming in VAX Fortran"
(1984 ed; I kept it because it makes a pretty good Fortran 77 reference):
the RECORDTYPE keyword to OPEN had 6 possible values (fixed, variable,
segmented, stream, stream_cr, stream_lf). As I recall, "variable", which
started each record with a 2-byte (4-byte?) length count, was pretty
common for text files, even more than the "stream" types. Then there was

ORGANIZATION which could be sequential, relative, or indexed...

And I think I read somewhere that Macs use (used?) just CR.

> If the LF and CR/LF are the only two, the only thing you'd have to do
> is counting the number of 10b in the byte-filed:
>
> f = read_binary('Big_honking_example_file')
> h = histogram(f)
> print,h[10]
> 1479054
>
> If you're really intent on accessing the individual data items in the
> file, you could retain the reverse indices of the histogram for a
> handy field of pointers to each individual line that can be converted
> into a string at will...

That's pretty cool, but wouldn't they point to the _ends_ of the lines?
I suppose you could add 1, pop off the last value, and push a 0 onto the
front of the array.

Jeff Guerber

Subject: Re: Read Total lines in an ASCII file
Posted by James Kuyper on Sat, 21 Dec 2002 21:03:14 GMT
View Forum Message <> Reply to Message

Big Bird wrote:
...
> not necessarily try to convert them into anything. The only deviation
> from the usual 10b linefeed out there on idl'ish platforms is the DOS
> [10b,13b] LF/CR, right? Or do VMS systems do yet something different?
> How do the various suggested methods hold up on VMS?

There's at least one platform (Mac?) where CR/LF is a newline, rather
than LF/CR. There are machines where text files are stored in
fixed-length records, with each line starting a new record, and padded
out to the end of the record with 0s or blanks (I don't remember which).

I have no idea whether IDL runs on any such machines.
