Subject: Re: Possible Mac OS Bug with FIX.
Posted by Foldy Lajos on Wed, 11 May 2005 17:04:11 GMT
View Forum Message <> Reply to Message

Hi David,

I guess this is an endianness thing. You are converting the 5 0 byte sequence to a 2-byte integer. It can be 5*256+0 (1280) or 5+0*256 (5) depending on the host endianness.

regards,
lajos

On Wed, 11 May 2005, David Fanning wrote:

> Folks,
>
> I have a report of a possible bug this morning with the Mac OS
> operating system. It seems that the FIX function, when used with
> the offset parameter does not produce reliable values.
>
> Here is an example:
>
> tmp=bytarr(20)
> tmp[15]=5
> print, fix(tmp[15]), fix(tmp,15)
> 5 1280
>
> Can anyone confirm that this is or is not a problem in
> IDL 6.1 on the Mac OS operating system?
>
> Thanks,
>
> David
>
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
>

Subject: Re: Possible Mac OS Bug with FIX.
Posted by K. Bowman on Wed, 11 May 2005 17:12:13 GMT
View Forum Message <> Reply to Message

In article <MPG.1cebe581b001a48e9899fc@news.frii.com>,
 David Fanning <davidf@dfanning.com> wrote:

> Folks,
>
> I have a report of a possible bug this morning with the Mac OS
> operating system. It seems that the FIX function, when used with
> the offset parameter does not produce reliable values.
>
> Here is an example:
>
>   tmp=bytarr(20)
>   tmp[15]=5
>   print, fix(tmp[15]), fix(tmp,15)
>   5   1280
>
> Can anyone confirm that this is or is not a problem in
> IDL 6.1 on the Mac OS operating system?
>
> Thanks,
>
> David

I think perhaps you want fix(tmp, 14)?

```
IDL> tmp = bytarr(20)
IDL> tmp[15] = 5
IDL> print, fix(tmp[15])
      5
IDL> print, fix(tmp, 14)
      5
```

I think it is not a question of endianess, but of "which 2 bytes belong to the
2-byte integer".

Ken

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 17:55:46 GMT
View Forum Message <> Reply to Message

Kenneth Bowman writes:

> I think perhaps you want fix(tmp, 14)?
>
> IDL> tmp = bytarr(20)
> IDL> tmp[15] = 5

> IDL> print, fix(tmp[15])
>   5
> IDL> print, fix(tmp, 14)
>   5
>
> I think it is not a question of endianess, but of "which 2 bytes belong to the
> 2-byte integer".

What we are really looking for is consistency across
architectures. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by Foldy Lajos on Wed, 11 May 2005 18:10:54 GMT
View Forum Message <> Reply to Message

Hi,

my guess is that the two fix() calls do different things:

- fix(tmp[15]) does convert a 1-byte integer to a 2-byte integer, using
  a conversion instruction in the CPU. The result is the same on all
  architectures.

- fix(tmp, 15) does read memory from the address of tmp[15] as a 2-byte
  integer. No conversion instruction is used, only write (as type A) and
  read (as type B). It is the good old EQUIVALENCE statement from Fortran,
  and is architecture-dependent. Perhaps this should be mentioned in the
  IDL docs :-)

regards,
lajos

On Wed, 11 May 2005, David Fanning wrote:

> Kenneth Bowman writes:
>
>> I think perhaps you want fix(tmp, 14)?

>>
>> IDL> tmp = bytarr(20)
>> IDL> tmp[15] = 5
>> IDL> print, fix(tmp[15])
>>   5
>> IDL> print, fix(tmp, 14)
>>   5
>>
>> I think it is not a question of endianess, but of "which 2 bytes belong to the
>> 2-byte integer".
>
> What we are really looking for is consistency across
> architectures. :-)
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
>

Subject: Re: Possible Mac OS Bug with FIX.
Posted by K. Bowman on Wed, 11 May 2005 18:13:11 GMT
View Forum Message <> Reply to Message

In article <MPG.1cebf50de6204f689899fd@news.frii.com>,
 David Fanning <davidf@dfanning.com> wrote:

> What we are really looking for is consistency across
> architectures. :-)

Ah, that would make it an RSI issue then. :-)

Perhaps they could also clarify whether OFFSET indicates the first or last byte
of the soon-to-be-different-type variable.

Ken

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 18:31:23 GMT
View Forum Message <> Reply to Message

=?ISO-8859-2?Q?F=F6ldy_Lajos?= writes:

> my guess is that the two fix() calls do different things:
>
> - fix(tmp[15]) does convert a 1-byte integer to a 2-byte integer, using
> a conversion instruction in the CPU. The result is the same on all
> architectures.
>
> - fix(tmp, 15) does read memory from the address of tmp[15] as a 2-byte
> integer. No conversion instruction is used, only write (as type A) and
> read (as type B). It is the good old EQUIVALENCE statement from Fortran,
> and is architecture-dependent. Perhaps this should be mentioned in the
> IDL docs :-)

Well, as I say, I understand the problem (and can even
appreciate it), but what I am after is consistency. So,
how architecture-dependent *is* it? Does the same thing
happen on Linux machines? I'm just trying to get a feel
for when I should put the fix in. :-)

One of my sources points out that the procedure
SWAP_ENDIAN_INPLACE is useful, with one of the SWAP_IF_****
keywords is used. I just want to know if I have to use
it only on Macs, or on UNIX machines, or when!?

Is a Macintosh a big or little endian machine?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by Foldy Lajos on Wed, 11 May 2005 18:45:37 GMT
View Forum Message <> Reply to Message

From Wikipedia (endianness):

To summarize, here is the default endian-formats of some common computer architectures:

* Pure big-endian: Sun SPARC, Motorola 68000, PowerPC 970, IBM System/360
* Bi-endian, running in big-endian mode by default: MIPS running IRIX, PA-RISC, most POWER
and PowerPC systems

* Bi-Endian, running in little-endian mode by default: MIPS running Ultrix, most DEC Alpha, IA-64 running Linux
* Pure little-endian: Intel x86, AMD64, DEC VAX (excluding D-Float numbers)

So x86/linux, x86/windoz is little-endian, Mac is big-endian.

regards,
lajos

On Wed, 11 May 2005, David Fanning wrote:

> Is a Macintosh a big or little endian machine?
> Cheers,
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by K. Bowman on Wed, 11 May 2005 18:53:30 GMT
View Forum Message <> Reply to Message

In article <MPG.1cebfd471f9e27a9899fe@news.frii.com>,
David Fanning <davidf@dfanning.com> wrote:

> One of my sources points out that the procedure
> SWAP_ENDIAN_INPLACE is useful, with one of the SWAP_IF_****
> keywords is used. I just want to know if I have to use
> it only on Macs, or on UNIX machines, or when!?
> Is a Macintosh a big or little endian machine?

Since these statements are running on the same machine, why would endianess matter? (Not a rhetorical question ... just confused like David.)

Ken Bowman

Subject: Re: Possible Mac OS Bug with FIX.

Posted by K. Bowman on Wed, 11 May 2005 18:58:49 GMT
View Forum Message <> Reply to Message

This may not be relevant, but I know that some compilers enforce alignment of
2-byte variables on 2-byte boundaries, 4-byte variables on 4-byte boundaries,
etc., while some do not.

Ken Bowman

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 19:30:12 GMT
View Forum Message <> Reply to Message

Kenneth Bowman writes:

> Ah, that would make it an RSI issue then. :-)

Well, of course. What didn't I think of that! :-)

I personally thing this is a programming error of the
sort you are not likely to think about until the missile
is WAY off course. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 19:37:10 GMT
View Forum Message <> Reply to Message

Kenneth Bowman writes:

> Since these statements are running on the same machine, why would endianess
> matter? (Not a rhetorical question ... just confused like David.)

Well, thank you! This is the dilemma in a nutshell.

You have a string of bytes, sequentially numbered, obviously.
If you are asked to read two of those sequential bytes and make
it an integer, wouldn't you expect the FIX programmer to treat

the first byte as the "lowest" bits (first in time, I guess)
and the second byte as the "highest" bits (next in time).
Then, depending upon which machine you are running on,
you would put the "lowest" bits first or second, depending
on endianess (is that a word?).

That obviously makes it an RSI problem. Or, would RSI say
I don't know what the hell you are thinking, YOU figure it
out?

I'd like to know before I place the phone call. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 19:56:25 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Kenneth Bowman writes:
>
>> Since these statements are running on the same machine, why would endianess
>> matter? (Not a rhetorical question ... just confused like David.)
>
> Well, thank you! This is the dilemma in a nutshell.
>
> You have a string of bytes, sequentially numbered, obviously.
> If you are asked to read two of those sequential bytes and make
> it an integer, wouldn't you expect the FIX programmer to treat
> the first byte as the "lowest" bits (first in time, I guess)
> and the second byte as the "highest" bits (next in time).
> Then, depending upon which machine you are running on,
> you would put the "lowest" bits first or second, depending
> on endianess (is that a word?).
>
> That obviously makes it an RSI problem. Or, would RSI say
> I don't know what the hell you are thinking, YOU figure it
> out?
>
> I'd like to know before I place the phone call. :-)

After thinking about it some more, I still think it
is the IDL programmer's job to know what he is doing.
You don't know where the bytes came from. (And what is the
endian nature of the machine *creating* the bytes. Oh, Lord,
do I have to worry about that, too!?)

Obviously, there are too many unanswered questions to expect
RSI to figure it out for you. If you are doing something like
this, I think the onus must be on you to figure it out.

What I am still not clear about, however, is whether whatever
the hell it is that RSI is doing is done *consistently*
across all machine architectures. Anyone have a theory about
that? :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.
Posted by savoie on Wed, 11 May 2005 20:14:41 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> writes:

> What I am still not clear about, however, is whether whatever
> the hell it is that RSI is doing is done *consistently*
> across all machine architectures. Anyone have a theory about
> that? :-)

It seems pretty clear to me what's happening is what lajos previously
stated. But the more I tried to articulate that, the less sense it made.
Anyway, on my two machines,

Linux little endian:

IDL> tmp = bytarr(20)
tmp = bytarr(20)

```
IDL> tmp[15] = 5
tmp[15] = 5
IDL> print, fix(tmp[15]), fix(tmp,15)
print, fix(tmp[15]), fix(tmp,15)
       5       5
```

On a Big Endian IRIX64 machine

```
---------------------------
IDL> tmp=bytarr(20)
IDL> tmp[15] = 5
IDL> print, fix(tmp[15]), fix(tmp,15)
       5    1280
```

I don't even know if this helps.

Matt

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

Subject: Re: Possible Mac OS Bug with FIX.
Posted by David Fanning on Wed, 11 May 2005 20:44:08 GMT
View Forum Message <> Reply to Message

savoie@nsidc.org writes:

> It seems pretty clear to me what's happening is what lajos previously
> stated. But the more I tried to articulate that, the less sense it made.

You don't know how happy it makes me to know I'm not
the only one confused by this. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Possible Mac OS Bug with FIX.

Posted by Dick Jackson on Wed, 11 May 2005 22:50:38 GMT
View Forum Message <> Reply to Message

<savoie@nsidc.org> wrote in message
news:ywkufywtwmdq.fsf@snowblower.colorado.edu...
> David Fanning <davidf@dfanning.com> writes:
>
>
>
>> What I am still not clear about, however, is whether whatever
>> the hell it is that RSI is doing is done *consistently*
>> across all machine architectures. Anyone have a theory about
>> that? :-)
>
> It seems pretty clear to me what's happening is what lajos previously
> stated. But the more I tried to articulate that, the less sense it made.
> Anyway, on my two machines,
>
>
> Linux little endian:
> ------------------
>
> IDL> tmp = bytarr(20)
> tmp = bytarr(20)
> IDL> tmp[15] = 5
> tmp[15] = 5
> IDL> print, fix(tmp[15]), fix(tmp,15)
> print, fix(tmp[15]), fix(tmp,15)
> 5 5
>
> On a Big Endian IRIX64 machine
> ----------------------------
> IDL> tmp=bytarr(20)
> IDL> tmp[15] = 5
> IDL> print, fix(tmp[15]), fix(tmp,15)
> 5 1280
>
>
> I don't even know if this helps.

I think you are exactly right, but to get David's consistency across
platforms:

```
IDL> tmp = bytarr(20)
IDL> tmp[15] = 5
IDL> result=fix(tmp,15)
IDL> print,result       ; This should vary across platforms
      5
IDL> swap_endian_inplace, result, /swap_if_little_endian
```

```
IDL> print,result       ; This should be consistent across platforms
   1280
```

(the reader will guess I'm on an Intel box, someone please test this elsewhere!)

Now, there are still two issues that are for David to decide:
- do you really want /swap_if_little_endian or /swap_if_big_endian?
  (either will give a consistent result)
- do you really want the bytes starting at an offset that is not a
  multiple of the size of the type you're casting it to (Fix: 2 bytes)?

Hope this helps!

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392