Subject: Re: Array Subscripting Memory Usage (watch out!)
Posted by R.Bauer on Fri, 04 Oct 2002 19:40:33 GMT
View Forum Message <> Reply to Message

Dick Jackson wrote:

```
> Hi all,
>
> This may be old news to some of you, but it surprised me and a couple of
> colleagues, and I couldn't find any discussion of it on this group, so
> I'll share it around.
>
> I was surprised to find how much memory is used during access to a
> subset of an array. I ran this, which makes a 1000x1000 array, and
> accesses a subset of it using an array of subscripts:
>
> a = bindgen(1000, 1000)
> subscripts = Long(RandomU(seed, 500)*1000)
> baseMem = (memory())[0]
> help, a[subscripts, *]
> highWaterMem = (memory())[3]
> Print, 'Memory used during access: ', highWaterMem-baseMem
>
> IDL> .GO
> <Expression>    BYTE      = Array[500, 1000]
> Memory used during access:      2500076
```

Dear all, especially David please can you put this on your tips pages.

My last posting wasn't enough detailed to describe what is going on.
So here is the next one.

a = bindgen(1000, 1000)
then size of a is 1 MB

If you now want to do something by subscribts the subscribts never passed by
reference they are passed by value of type of the indices.

For example:
help,a[indgen(10)] takes during execution 50*2+ 50 bytes.

and
help,a[indgen(10,/long)] takes during execution 50*4 +50 bytes

Now what happens if we set a *. This means nothing else as from start to end
and it is an alias for an index array of type long.

help,a[subscripts,*]

is the same as help, a[subscripts[0:499],0:999]

this means the subscripts of a are
500 x 1000 x 4 and this is 2 MB

plus the result after the index operation in byte of 500 x 1000 which gives
0.5MB is the measured 2.5 MB.

During the operation you need this amount of memory.
All the array functionalities of idl goes by cost of memory.

Now something about byte. The byte or string isn't handled the same as the others.
I don't know why. May be they are internal first defined in something different.

That's for a new discussion:

BYTE:

```
.reset
s=memory()&x=indgen(20,/byte)&e=memory()& print,e[3]-s[0]
     188
.reset
s=memory()&x=indgen(10,/byte)&e=memory()& print,e[3]-s[0]
     180
```

The difference is 8 normally it should be 10 ?!:-(
where are the two bytes?

```
.reset
s=memory()&x=indgen(8,/byte)&e=memory()& print,e[3]-s[0]
     176

.reset
s=memory()&x=indgen(4,/byte)&e=memory()& print,e[3]-s[0]
     172
```

In this example 4 byte less is 4 byte less memory.

.reset

```
s=memory()&x=indgen(2,/byte)&e=memory()& print,e[3]-s[0]
     172
```
And 2 byte less then 4 byte costs the same memory as 4 byte.
Did this mean 2 byte costs nothing?

STRING:
```
.reset
s=memory()&x=indgen(20,/string)&e=memory()& print,e[3]-s[0]
     748
.reset
s=memory()&x=indgen(10,/string)&e=memory()& print,e[3]-s[0]
     458
```
The difference is 290. How to explain, the string length is 12 of each
element. Normally I believe 1 char is 1 byte. So first result should be
12 byte * 20 = 240 and the second one 120. The differnce should be 120.
So I have no explanation why it needs so much more memory. Any ideas?

The numbers goes well.

INTEGER:
```
.reset
s=memory()&x=indgen(20)&e=memory()& print,e[3]-s[0]
     208
.reset
s=memory()&x=indgen(10)&e=memory()& print,e[3]-s[0]
     188
```

The difference is exactly 20 because integer is 2 byte and
first 10 values more.

LONG:
```
.reset
s=memory()&x=indgen(20,/long)&e=memory()& print,e[3]-s[0]
     248
.reset
s=memory()&x=indgen(10,/long)&e=memory()& print,e[3]-s[0]
     208
```

The difference is 40 and this is correct too because long is
4 byte.

FLOAT:
```
.reset
s=memory()&x=indgen(20,/float)&e=memory()& print,e[3]-s[0]
     248
```

```
.reset
s=memory()&x=indgen(10,/float)&e=memory()& print,e[3]-s[0]
      208
```

A float number has the same size as a long number so this is correct too.

Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 == ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html