GammaLib - Bug #589

Run Valgrind on unit tests?

11/07/2012 01:30 PM - Deil Christoph

Status:	New	**Start date:**	11/07/2012
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I just tried running the api-sanity-checker on my Mac against gammalib built with clang.

Compared to the results at https://cta-jenkins.irap.omp.eu/job/gammalib-sanity/ there's an additional segfault in the `GIntegral::polint` test:

```
#include <stdlib.h>
#include <gammalib/GammaLib.hpp>

class GIntegrand_SubClass: public GIntegrand
{
public:
   GIntegrand_SubClass():GIntegrand(){}

   double eval(double x) {
     return 6.5;
   }
};//GIntegrand_SubClass

class GIntegral_SubClass: public GIntegral
{
public:
   GIntegral_SubClass(GIntegrand* integrand):GIntegral(integrand){}

   double polint_Wrapper() {
     double* xa = (double*) malloc(sizeof(double)*256);
     double ya [] = {
         1.5,
         2.5,
         3.5,
         4.5};
     double* dy = (double*) malloc(sizeof(double)*256);
     return this->polint(
        xa,
        ya,
        256,
        5.5,
        dy);
   }
};//GIntegral_SubClass

int main(int argc, char *argv[])
{
   GIntegrand_SubClass* integrand = new GIntegrand_SubClass();
   GIntegral_SubClass* integral = new GIntegral_SubClass(integrand);
   integral->polint_Wrapper(); //target call
   return 0;
}
```

I didn't investigate what the actual problem is, here's what Valgrind has to say (on Ubuntu, Valgrind is pretty broken on Mac OS X

10.8):

```
deil@ubuntu:~/software/code/gammalib_sanity/tests/libgamma/0.0/groups/GIntegral/classes/GIntegral/_ZN9GIntegral6polintEPdS0_idS0_$ valgrind ./test | grep -v 'are identical'
==34252== Memcheck, a memory error detector
==34252== Copyright (C) 2002-2011, and GNU GPL'd, by Julian Seward et al.
==34252== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==34252== Command: ./test
==34252==
==34252== Conditional jump or move depends on uninitialised value(s)
==34252==    at 0x4EFAE19: GIntegral::polint(double*, double*, int, double, double*) (GIntegral.cpp:468)
==34252==    by 0x400EF9: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==
==34252== Invalid read of size 8
==34252==    at 0x4EFAE0B: GIntegral::polint(double*, double*, int, double, double*) (GIntegral.cpp:467)
==34252==    by 0x400EF9: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==  Address 0x6efa540 is 0 bytes after a block of size 2,048 alloc'd
==34252==    at 0x4C2B6CD: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==34252==    by 0x400E87: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==
==34252== Conditional jump or move depends on uninitialised value(s)
==34252==    at 0x4EFAF57: GIntegral::polint(double*, double*, int, double, double*) (GIntegral.cpp:489)
==34252==    by 0x400EF9: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==
==34252== Conditional jump or move depends on uninitialised value(s)
==34252==    at 0x4EFAF5D: GIntegral::polint(double*, double*, int, double, double*) (GIntegral.cpp:489)
==34252==    by 0x400EF9: GIntegral_SubClass::polint_Wrapper() (in
```

```
==34252==    by 0x400D6A: main (in

==34252==
==34252== Invalid read of size 8
==34252==    at 0x4EFAF20: GIntegral::polint(double*, double*, int, double, double*) (GIntegral.cpp:486)
==34252==    by 0x400EF9: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==  Address 0x6efa540 is 0 bytes after a block of size 2,048 alloc'd
==34252==    at 0x4C2B6CD: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==34252==    by 0x400E87: GIntegral_SubClass::polint_Wrapper() (in

==34252==    by 0x400D6A: main (in

==34252==
==34252==
==34252== HEAP SUMMARY:
==34252==     in use at exit: 6,040 bytes in 58 blocks
==34252==   total heap usage: 219 allocs, 161 frees, 21,263 bytes allocated
==34252==
==34252== LEAK SUMMARY:
==34252==    definitely lost: 4,136 bytes in 3 blocks
==34252==    indirectly lost: 8 bytes in 1 blocks
==34252==      possibly lost: 1,640 bytes in 46 blocks
==34252==    still reachable: 256 bytes in 8 blocks
==34252==         suppressed: 0 bytes in 0 blocks
==34252== Rerun with --leak-check=full to see details of leaked memory
==34252==
==34252== For counts of detected and suppressed errors, rerun with: -v
==34252== Use --track-origins=yes to see where uninitialised values come from
==34252== ERROR SUMMARY: 65792 errors from 5 contexts (suppressed: 2 from 2)
```

So the api-sanity-checker only runs the tests and looks for segfaults, but really we want to check for memory errors and it's random if those trigger segfaults or not.

Should I write a small script to run valgrind on all tests (not only the api-sanity-checker) and summarizes the results?
(I thought I had seen valgrind already on your Jenkins, but now I can't find it!?)

History

#1 - 11/15/2012 09:17 AM - Deil Christoph

A few api-sanity-checker bugs have been fixed in the last weeks in github master based on my feedback for gammalib:
https://github.com/lvc/api-sanity-checker/issues
TODO: update Jenkins api-sanity-checker when they make the new release (should be soon), it will generate better tests.

I've asked if they could add a `run-valgrind` option:
https://github.com/lvc/api-sanity-checker/issues/9

The api-sanity-checker dev commented that everything looks good with gammalib, except for this OpenMP linking issue:
https://github.com/lvc/api-sanity-checker/issues/7#issuecomment-10400068

#2 - 11/30/2012 09:49 PM - Knödlseder Jürgen

Thanks for doing the tests. The OpenMP linking issue seems to indicate that the -lgomp option is not needed, but I'm not sure to which Makefile this refers. In GammaLib-00-06-01 this option was indeed added in linking the library when built on Darwin, but since GammaLib-00-06-02 this option does not exist anymore. Which version have you tested?

It seems that the last version of the sanity check is still 1.12.10, that's the version that is actually installed on Jenkins.

#3 - 12/05/2012 11:04 AM - Deil Christoph

Jürgen Knödlseder wrote:

> Thanks for doing the tests. The OpenMP linking issue seems to indicate that the -lgomp option is not needed, but I'm not sure to which Makefile this refers. In GammaLib-00-06-01 this option was indeed added in linking the library when built on Darwin, but since GammaLib-00-06-02 this option does not exist anymore. Which version have you tested?

Actually I don't know which gammalib version aponomarenko refers to here about the OpenMP linking issue:
https://github.com/lvc/api-sanity-checker/issues/7#issuecomment-10400068

I just tested git devel and there libgamma correctly says libgomp is needed:

```
deil@ubuntu-virtualbox:~/software/gammalib$ readelf -Wa src/.libs/libgamma.so | grep NEEDED
 0x00000001 (NEEDED)                     Shared library: [libcfitsio.so.3]
 0x00000001 (NEEDED)                     Shared library: [libm.so.6]
 0x00000001 (NEEDED)                     Shared library: [libgomp.so.1]
 0x00000001 (NEEDED)                     Shared library: [libgcc_s.so.1]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
deil@ubuntu-virtualbox:~/software/gammalib$ readelf -Wa src/.libs/libgamma.so | grep GOMP_
00265080  00000307 R_386_JUMP_SLOT       00000000   GOMP_sections_end
002654e4  00001b07 R_386_JUMP_SLOT       00000000   GOMP_critical_start
002657fc  00003207 R_386_JUMP_SLOT       00000000   GOMP_sections_start
002660a0  00006507 R_386_JUMP_SLOT       00000000   GOMP_barrier
002662ac  00007307 R_386_JUMP_SLOT       00000000   GOMP_sections_next
0026678c  00008807 R_386_JUMP_SLOT       00000000   GOMP_parallel_end
00266b94  0000a207 R_386_JUMP_SLOT       00000000   GOMP_critical_end
0026712c  0000b907 R_386_JUMP_SLOT       00000000   GOMP_parallel_start
     3: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_end@GOMP_1.0 (4)
    27: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_critical_start@GOMP_1.0 (4)
    50: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_start@GOMP_1.0 (4)
   101: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_barrier@GOMP_1.0 (4)
   115: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_next@GOMP_1.0 (4)
   136: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_parallel_end@GOMP_1.0 (4)
   162: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_critical_end@GOMP_1.0 (4)
   185: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_parallel_start@GOMP_1.0 (4)
   878: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_end@@GOMP_1.0
  1672: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_critical_start@@GOMP_1.0
  2256: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_start@@GOMP_1.0
  3692: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_barrier@@GOMP_1.0
  4052: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_sections_next@@GOMP_1.0
  4946: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_parallel_end@@GOMP_1.0
  5684: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_critical_end@@GOMP_1.0
  6684: 00000000     0 FUNC    GLOBAL DEFAULT  UND GOMP_parallel_start@@GOMP_1.0
  000:   0 (*local*)       2 (GLIBC_2.0)     3 (GLIBC_2.0)     4 (GOMP_1.0)
  018:   0 (*local*)       0 (*local*)       2 (GLIBC_2.0)     4 (GOMP_1.0)
  030:   0 (*local*)       0 (*local*)       4 (GOMP_1.0)      0 (*local*)
  064:   a (GLIBC_2.1)     4 (GOMP_1.0)      0 (*local*)       0 (*local*)
  070:   0 (*local*)       3 (GLIBC_2.0)     0 (*local*)       4 (GOMP_1.0)
  088:   4 (GOMP_1.0)      2 (GLIBC_2.0)     2 (GLIBC_2.0)     0 (*local*)
  0a0:   0 (*local*)       0 (*local*)       4 (GOMP_1.0)      0 (*local*)
  0b8:   0 (*local*)       4 (GOMP_1.0)      0 (*local*)       3 (GLIBC_2.0)
  0x0070:   Name: GOMP_1.0  Flags: none  Version: 4
```

It seems that the last version of the sanity check is still 1.12.10, that's the version that is actually installed on Jenkins.

You can just use git master for the api-sanity-checker, but this is not urgent, so you could just wait for the new version.
I think gammalib is pretty sane already :-)