GammaLib - Bug #1451

adding GCTABackground3D to GCTAObservation causes segfault

03/25/2015 10:33 PM - Kelley-Hoskins Nathan

Status:	Closed	**Start date:**	03/25/2015
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

After attempting to add a GCTABackground3D() object as the background to the GCTAObservation returned from obsutils.set_obs(), my python script segfaults. I simplfied my code to the below script, where commenting out the 'run.response()...' line removes the segfault.

backgroundbug.py:

```
#!/usr/bin/python
#import ROOT
import gammalib
from ctools import obsutils
back = gammalib.GCTABackground3D()
pntdir = gammalib.GSkyDir()
pntdir.radec( 30.0, 30.0 )
run = obsutils.set_obs( pntdir )
run.response().background(back)
```

output:

```
[1]    25998 segmentation fault  ./backgroundbug.py
```

output from the same script, importing root first:

```
 *** Break *** segmentation violation

==========================================================
There was a crash.
This is the entire stack trace of all threads:
==========================================================
#0  0x00000030bb8ac61e in waitpid () from /lib64/libc.so.6
#1  0x00000030bb83e609 in do_system () from /lib64/libc.so.6
#2  0x00007f768bdf8668 in TUnixSystem::StackTrace() () from /afs/ifh.de/group/cta/cta/software/root/root_v5.34.14/lib/libCore.so
#3  0x00007f768bdf74e3 in TUnixSystem::DispatchSignals(ESignals) () from
 /afs/ifh.de/group/cta/cta/software/root/root_v5.34.14/lib/libCore.so
#4  <signal handler called>
#5  0x00000030bbb902e8 in main_arena () from /lib64/libc.so.6
#6  0x00007f7686e0d24f in _wrap_delete_GCTABackground3D (args=<value optimized out>) at gammalib/cta_wrap.cpp:53139
#7  0x00000030cf043c63 in PyObject_Call () from /usr/lib64/libpython2.6.so.1.0
#8  0x00000030cf043f38 in PyObject_CallFunctionObjArgs () from /usr/lib64/libpython2.6.so.1.0
#9  0x00007f7686db0ab5 in SwigPyObject_dealloc (v=0x1725d50) at gammalib/cta_wrap.cpp:1580
#10 0x00000030cf079e4b in ?? () from /usr/lib64/libpython2.6.so.1.0
#11 0x00000030cf09a75c in ?? () from /usr/lib64/libpython2.6.so.1.0
#12 0x00000030cf078287 in ?? () from /usr/lib64/libpython2.6.so.1.0
#13 0x00000030cf07acf7 in PyDict_SetItem () from /usr/lib64/libpython2.6.so.1.0
#14 0x00000030cf07c66d in _PyModule_Clear () from /usr/lib64/libpython2.6.so.1.0
#15 0x00000030cf0e987f in PyImport_Cleanup () from /usr/lib64/libpython2.6.so.1.0
```

#16 0x00000030cf0f28ab in Py_Finalize () from /usr/lib64/libpython2.6.so.1.0
#17 0x00000030cf0ff2d6 in Py_Main () from /usr/lib64/libpython2.6.so.1.0
#18 0x00000030bb81ed5d in __libc_start_main () from /lib64/libc.so.6
#19 0x0000000000400649 in _start ()
==========

The lines below might hint at the cause of the crash.
If they do not help you then please submit a bug report at
http://root.cern.ch/bugs. Please post the ENTIRE stack trace
from above as an attachment in addition to anything else
that might help us fixing this issue.
==========
#5 0x00000030bbb902e8 in main_arena () from /lib64/libc.so.6
#6 0x00007f7686e0d24f in _wrap_delete_GCTABackground3D (args=<value optimized out>) at gammalib/cta_wrap.cpp:53139
#7 0x00000030cf043c63 in PyObject_Call () from /usr/lib64/libpython2.6.so.1.0
#8 0x00000030cf043f38 in PyObject_CallFunctionObjArgs () from/usr/lib64/libpython2.6.so.1.0
#9 0x00007f7686db0ab5 in SwigPyObject_dealloc (v=0x1725d50) at gammalib/cta_wrap.cpp:1580
#10 0x00000030cf079e4b in ?? () from /usr/lib64/libpython2.6.so.1.0
#11 0x00000030cf09a75c in ?? () from /usr/lib64/libpython2.6.so.1.0
#12 0x00000030cf078287 in ?? () from /usr/lib64/libpython2.6.so.1.0
#13 0x00000030cf07acf7 in PyDict_SetItem () from /usr/lib64/libpython2.6.so.1.0
#14 0x00000030cf07c66d in _PyModule_Clear () from /usr/lib64/libpython2.6.so.1.0
#15 0x00000030cf0e987f in PyImport_Cleanup () from /usr/lib64/libpython2.6.so.1.0
#16 0x00000030cf0f28ab in Py_Finalize () from /usr/lib64/libpython2.6.so.1.0
#17 0x00000030cf0ff2d6 in Py_Main () from /usr/lib64/libpython2.6.so.1.0
#18 0x00000030bb81ed5d in __libc_start_main () from /lib64/libc.so.6
#19 0x0000000000400649 in _start ()
==========

my original code is longer and uglier, but its segfault stacktrace was slightly different:

*** Break *** segmentation violation

==========
There was a crash.
This is the entire stack trace of all threads:
==========

Thread 2 (Thread 0x7f04c71a7700 (LWP 24207)):
#0 0x00000030bc40d930 in sem_wait () from /lib64/libpthread.so.0
#1 0x00000030cf0fd438 in PyThread_acquire_lock () from /usr/lib64/libpython2.6.so.1.0
#2 0x00000030cf0d7794 in PyEval_RestoreThread () from /usr/lib64/libpython2.6.so.1.0
#3 0x00007f04d3a2d22f in ?? () from /usr/lib64/python2.6/lib-dynload/timemodule.so
#4 0x00000030cf0d59e4 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#5 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#6 0x00000030cf06adad in ?? () from /usr/lib64/libpython2.6.so.1.0
#7 0x00000030cf043c63 in PyObject_Call () from /usr/lib64/libpython2.6.so.1.0
#8 0x00000030cf0d4470 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#9 0x00000030cf0d6b8f in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#10 0x00000030cf0d6b8f in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#11 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#12 0x00000030cf06acb0 in ?? () from /usr/lib64/libpython2.6.so.1.0
#13 0x00000030cf043c63 in PyObject_Call () from /usr/lib64/libpython2.6.so.1.0
#14 0x00000030cf0566af in ?? () from /usr/lib64/libpython2.6.so.1.0
#15 0x00000030cf043c63 in PyObject_Call () from /usr/lib64/libpython2.6.so.1.0
#16 0x00000030cf0cfc93 in PyEval_CallObjectWithKeywords () from /usr/lib64/libpython2.6.so.1.0
#17 0x00000030cf1017ba in ?? () from /usr/lib64/libpython2.6.so.1.0
#18 0x00000030bc4079d1 in start_thread () from /lib64/libpthread.so.0
#19 0x00000030bb8e88fd in clone () from /lib64/libc.so.6

Thread 1 (Thread 0x7f04d9d3d700 (LWP 24199)):
#0 0x00000030bb8ac65d in waitpid () from /lib64/libc.so.6
#1 0x00000030bb83e609 in do_system () from /lib64/libc.so.6
#2 0x00000030bb83e940 in system () from /lib64/libc.so.6

```
#3  0x00007f04c9622668 in TUnixSystem::StackTrace() () from /afs/ifh.de/group/cta/cta/software/root/root_v5.34.14/lib/libCore.so
#4  0x00007f04c96214e3 in TUnixSystem::DispatchSignals(ESignals) () from
 /afs/ifh.de/group/cta/cta/software/root/root_v5.34.14/lib/libCore.so
#5  <signal handler called>
#6  0x0000000003af7be0 in ?? ()
#7  0x00007f04ce20d6cc in GCTAResponseIrf::copy_members (this=0x3afef90, rsp=...) at src/GCTAResponseIrf.cpp:3033
#8  0x00007f04ce20dc68 in GCTAResponseIrf::GCTAResponseIrf (this=0x3afef90, rsp=...) at src/GCTAResponseIrf.cpp:150
#9  0x00007f04ce20dd49 in GCTAResponseIrf::clone (this=0x560a500) at src/GCTAResponseIrf.cpp:278
#10 0x00007f04ce1e9e57 in GCTAObservation::copy_members (this=0x55ee060, obs=...) at src/GCTAObservation.cpp:1325
#11 0x00007f04ce1ea39e in GCTAObservation::GCTAObservation (this=0x55ee060, obs=...) at src/GCTAObservation.cpp:158
#12 0x00007f04ce1ea429 in GCTAObservation::clone (this=0x55ef4a0) at src/GCTAObservation.cpp:246
#13 0x00007f04ce1421e9 in GObservations::append (this=0x48e9bd0, obs=...) at GObservations.cpp:316
#14 0x00007f04ccc96e1b in _wrap_GObservations_append (args=<value optimized out>) at gammalib/obs_wrap.cpp:19663
#15 0x00000030cf0d55c6 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#16 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#17 0x00000030cf0d5aa4 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#18 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#19 0x00000030cf0d5aa4 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#20 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#21 0x00000030cf0d7732 in PyEval_EvalCode () from /usr/lib64/libpython2.6.so.1.0
#22 0x00000030cf0f1bac in ?? () from /usr/lib64/libpython2.6.so.1.0
#23 0x00000030cf0f1c80 in PyRun_FileExFlags () from /usr/lib64/libpython2.6.so.1.0
#24 0x00000030cf0f316c in PyRun_SimpleFileExFlags () from /usr/lib64/libpython2.6.so.1.0
#25 0x00000030cf0ff8a2 in Py_Main () from /usr/lib64/libpython2.6.so.1.0
#26 0x00000030bb81ed5d in __libc_start_main () from /lib64/libc.so.6
#27 0x0000000000400649 in _start ()
==========================================================

The lines below might hint at the cause of the crash.
If they do not help you then please submit a bug report at
http://root.cern.ch/bugs. Please post the ENTIRE stack trace
from above as an attachment in addition to anything else
that might help us fixing this issue.
==========================================================
#6  0x0000000003af7be0 in ?? ()
#7  0x00007f04ce20d6cc in GCTAResponseIrf::copy_members (this=0x3afef90, rsp=...) at src/GCTAResponseIrf.cpp:3033
#8  0x00007f04ce20dc68 in GCTAResponseIrf::GCTAResponseIrf (this=0x3afef90, rsp=...) at src/GCTAResponseIrf.cpp:150
#9  0x00007f04ce20dd49 in GCTAResponseIrf::clone (this=0x560a500) at src/GCTAResponseIrf.cpp:278
#10 0x00007f04ce1e9e57 in GCTAObservation::copy_members (this=0x55ee060, obs=...) at src/GCTAObservation.cpp:1325
#11 0x00007f04ce1ea39e in GCTAObservation::GCTAObservation (this=0x55ee060, obs=...) at src/GCTAObservation.cpp:158
#12 0x00007f04ce1ea429 in GCTAObservation::clone (this=0x55ef4a0) at src/GCTAObservation.cpp:246
#13 0x00007f04ce1421e9 in GObservations::append (this=0x48e9bd0, obs=...) at GObservations.cpp:316
#14 0x00007f04ccc96e1b in _wrap_GObservations_append (args=<value optimized out>) at gammalib/obs_wrap.cpp:19663
#15 0x00000030cf0d55c6 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#16 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#17 0x00000030cf0d5aa4 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#18 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#19 0x00000030cf0d5aa4 in PyEval_EvalFrameEx () from /usr/lib64/libpython2.6.so.1.0
#20 0x00000030cf0d7657 in PyEval_EvalCodeEx () from /usr/lib64/libpython2.6.so.1.0
#21 0x00000030cf0d7732 in PyEval_EvalCode () from /usr/lib64/libpython2.6.so.1.0
#22 0x00000030cf0f1bac in ?? () from /usr/lib64/libpython2.6.so.1.0
#23 0x00000030cf0f1c80 in PyRun_FileExFlags () from /usr/lib64/libpython2.6.so.1.0
#24 0x00000030cf0f316c in PyRun_SimpleFileExFlags () from /usr/lib64/libpython2.6.so.1.0
#25 0x00000030cf0ff8a2 in Py_Main () from /usr/lib64/libpython2.6.so.1.0
#26 0x00000030bb81ed5d in __libc_start_main () from /lib64/libc.so.6
#27 0x0000000000400649 in _start ()
==========================================================
```

History

#1 - 03/26/2015 12:57 AM - Knödlseder Jürgen

There is in fact some problem with GCTABackground3D(). The following code segfaults at the print(back) line:

```
#!/usr/bin/python
import gammalib
from ctools import obsutils
back = gammalib.GCTABackground3D()
print(back)
pntdir = gammalib.GSkyDir()
pntdir.radec( 30.0, 30.0 )
run = obsutils.set_obs( pntdir )
print(run)
print("Now")
print(back)
print("... do it")
run.response().background(back)
```

#2 - 03/26/2015 09:14 AM - Mayer Michael

It seems the problem occurs in GCTABackground3D::print(), where we try to print properties about the GCTAResponseTable. The segfault also occurs using the following code (which is called in GCTABackground3D::print):

```
import gammalib
rsp = gammalib.GCTAResponseTable()
rsp.axis_lo(0,0)
```

in rsp.axis_lo(0,0), we are trying to access an element of an empty vector -> segfault. Probably GCTAResponseTable::axis_lo() should have a check if the vector is empty (or GCTABackground::print should do this check before calling this function).

#3 - 04/09/2015 09:37 PM - Knödlseder Jürgen

- Project changed from ctools to GammaLib

#4 - 04/09/2015 10:41 PM - Knödlseder Jürgen

- Status changed from New to Feedback

- Assigned To set to Knödlseder Jürgen

- Target version set to 1.0.0

- % Done changed from 0 to 100

Although range checking was present in the GCTAResponseTable::axis_lo method it did not work as expected. As a C++ rule I learned that the expression

(index < 0 || index >= axes() || bin < 0 || bin >= axis[index].size())

is evaluated from left to right, hence for axes()=0 this should never reach the axis[index].size() expression. This obviously was not the case. Splitting the test into two independent test solved the issue.

So as a **big warning**: do not trust the evaluation order of logical or!

Furthermore a problem occurred in Python as the GCTAResponseIrf::background() method simply stored a pointer, hence the method deleted an element which still should have been alive in Python. This was prevented by cloning now the background in the GCTAResponseIrf::background() method, so that a copy now lives within GCTAResponseIrf over which the class has full control over. The same modification was also made for the other response components.

Finally, the GCTABackground3D method was corrected so that it also operated correctly on empty objects. Before some of the methods assumed that the background was loaded before, hence calling for example the operation() would leave to an exception. Now, the is_valid() method has been added to properly handle undefined objects.

#5 - 06/30/2015 12:49 PM - Kelley-Hoskins Nathan

I've tested this again with the fix applied, and it works fine. Would consider this issue closed.

#6 - 07/01/2015 02:58 PM - Knödlseder Jürgen

- Status changed from Feedback to Closed