ctools - Bug #1835

ctbin::save() applied on empty tool segfaults

08/03/2016 09:41 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	08/03/2016
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.2.0		

Description

The following Python codes segfaults:

```
bin = ctools.ctbin()
bin['debug']   = True
bin['outcube'] = 'ctbin_py0.fits'
bin.save()          # <== Segfaults
```

and

```
bin = ctools.ctbin()
bin['debug']   = True
bin['outcube'] = 'ctbin_py0.fits'
bin.publish()          # <== Segfaults
```

History

#1 - 08/03/2016 09:46 AM - Knödlseder Jürgen

And after a successful run, the following Python code leads to an exception:

```
bin.clear()         # <== Segfaults
libc++abi.dylib: terminating with uncaught exception of type GException::par_file_open_error: *** ERROR in GApplicationPars::write(std::string&):
 Unable to open parameter file "pfiles/".
Test ctbin from Python: ...../test_python_ctools.sh: line 25: 93462 Abort trap: 6           ./test_
```

#2 - 08/03/2016 09:47 AM - Knödlseder Jürgen

Check test_ctbin.py to reproduce the problems.

#3 - 08/04/2016 09:34 AM - Knödlseder Jürgen

A similar problem seems to exist with ctcubemask. The following code produces an exception:

```
mask.clear()
libc++abi.dylib: terminating with uncaught exception of type GException::par_file_open_error: *** ERROR in GApplicationPars::write(std::string&):
```

```
 Unable to open parameter file "pfiles/".
Test ctcubemask from Python: .....F./test_python_ctools.sh: line 25: 72729 Abort trap: 6           ./test_python_ctools.py
```

The stack trace of the exception is

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_kernel.dylib          0x00007fff8f553f06 __pthread_kill + 10
1   libsystem_pthread.dylib         0x00007fff9a27d4ec pthread_kill + 90
2   libsystem_c.dylib               0x00007fff955d46df abort + 129
3   libc++abi.dylib                 0x00007fff8ca57c11 abort_message + 257
4   libc++abi.dylib                 0x00007fff8ca7ddff default_terminate_handler() + 243
5   libobjc.A.dylib                 0x00007fff962e86c3 _objc_terminate() + 124
6   libc++abi.dylib                 0x00007fff8ca7b00e std::__terminate(void (*)()) + 8
7   libc++abi.dylib                 0x00007fff8ca7b083 std::terminate() + 51
8   libctools.2.dylib               0x000000010d2710af ctcubemask::~ctcubemask() + 15 (ctcubemask.cpp:128)
9   _tools.so                       0x000000010d205a56 _wrap_delete_ctcubemask(_object*, _object*) + 102 (tools_wrap.cpp:1447)
10  org.python.python               0x000000010b579eb0 PyObject_Call + 99
11  org.python.python               0x000000010b57648c PyObject_CallFunctionObjArgs + 190
12  _tools.so                       0x000000010d20da4d SwigPyObject_dealloc(_object*) + 445 (tools_wrap.cpp:1721)
13  org.python.python               0x000000010b5a5567 dict_dealloc + 129
14  org.python.python               0x000000010b5c586c subtype_dealloc + 575
15  org.python.python               0x000000010b5965c2 frame_dealloc + 110
16  org.python.python               0x000000010b5f32c1 PyEval_EvalCodeEx + 2047
17  org.python.python               0x000000010b597fb1 function_call + 352
18  org.python.python               0x000000010b579eb0 PyObject_Call + 99
19  org.python.python               0x000000010b584cb8 instancemethod_call + 173
20  org.python.python               0x000000010b579eb0 PyObject_Call + 99
21  org.python.python               0x000000010b5fd06f PyEval_CallObjectWithKeywords + 165
22  _test.so                        0x000000010cc6244e GPythonTestSuite::test() + 110 (test_wrap.cpp:3892)
23  libgamma.2.dylib                0x000000010bcdddd2 GTestSuite::run() + 674 (GTestSuite.cpp:319)
24  libgamma.2.dylib                0x000000010bce1ac9 GTestSuites::run() + 681 (GTestSuites.cpp:461)
25  _test.so                        0x000000010cc51f5a _wrap_GTestSuites_run(_object*, _object*) + 90 (test_wrap.cpp:3443)
26  org.python.python               0x000000010b5f9bcd PyEval_EvalFrameEx + 26858
27  org.python.python               0x000000010b5fd72d fast_function + 264
28  org.python.python               0x000000010b5f9af3 PyEval_EvalFrameEx + 26640
29  org.python.python               0x000000010b5f30f1 PyEval_EvalCodeEx + 1583
30  org.python.python               0x000000010b5fd69a fast_function + 117
31  org.python.python               0x000000010b5f9af3 PyEval_EvalFrameEx + 26640
32  org.python.python               0x000000010b5f30f1 PyEval_EvalCodeEx + 1583
33  org.python.python               0x000000010b5f2abc PyEval_EvalCode + 54
34  org.python.python               0x000000010b616ea1 run_mod + 53
35  org.python.python               0x000000010b616f44 PyRun_FileExFlags + 133
36  org.python.python               0x000000010b616a93 PyRun_SimpleFileExFlags + 698
37  org.python.python               0x000000010b628445 Py_Main + 3137
38  libdyld.dylib                   0x00007fff857105ad start + 1
```

From the stack trace the problem seems to occur in

```
ctbkgcube::~ctbkgcube(void)
{
    // Free members
    free_members();

    // Return
    return;
}
```

but this method does nothing. The exception method indicates that this has to do with writing the parameter file. It appears that the parameter file name is in fact empty (see the file name pfiles/ in the exception message). This happens after a call to clear(). The parameter file name is only set in a GApplication constructor, but this is not called in the clear() method. Probably the GApplication class should make sure that a number of attributes are preserved in a call to clear().

#4 - 08/04/2016 10:21 AM - Knödlseder Jürgen

I added some code to GApplication::clear() to preserve the name and version of the tool. Now all ctools need the following code in their clear method:

```
void ctcubemask::clear(void)
{
    // Free members
    free_members();
    this->ctool::free_members();

    // Clear base class (needed to conserve tool name and version)
    this->GApplication::clear();

    // Initialise members
    this->ctool::init_members();
    init_members();

    // Write header into logger
    log_header();

    // Return
    return;
}
```

#5 - 08/04/2016 10:57 AM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Knödlseder Jürgen

- Target version set to 1.2.0

- % Done changed from 0 to 50

The problems with calling the clear() method has been solved, yet the problem with the ctbin::save() and ctbin::publish() methods persist:

```
 *** Break *** segmentation violation
Test ctbin from Python: .. Generating stack trace...
 0x000000010d9e6bcd in PyEval_EvalFrameEx (in Python) + 26858
 0x000000010d9ea72d in fast_function (in Python) + 264
 0x000000010d9e6af3 in PyEval_EvalFrameEx (in Python) + 26640
 0x000000010d9e00f1 in PyEval_EvalCodeEx (in Python) + 1583
 0x000000010d984fb1 in function_call (in Python) + 352
 0x000000010d966eb0 in PyObject_Call (in Python) + 99
 0x000000010d971cb8 in instancemethod_call (in Python) + 173
 0x000000010d966eb0 in PyObject_Call (in Python) + 99
```

```
0x000000010d9ea06f in PyEval_CallObjectWithKeywords (in Python) + 165
0x000000010f04d44e in GPythonTestSuite::test() (in _test.so) (test_wrap.cpp:3892)
0x000000010e0c8d02 in GTestSuite::run() (in libgamma.2.dylib) (GTestSuite.cpp:319)
0x000000010e0cc9f9 in GTestSuites::run() (in libgamma.2.dylib) (GTestSuites.cpp:461)
0x000000010f03cf5a in _wrap_GTestSuites_run(_object*, _object*) (in _test.so) (test_wrap.cpp:3443)
0x000000010d9e6bcd in PyEval_EvalFrameEx (in Python) + 26858
0x000000010d9ea72d in fast_function (in Python) + 264
0x000000010d9e6af3 in PyEval_EvalFrameEx (in Python) + 26640
0x000000010d9e00f1 in PyEval_EvalCodeEx (in Python) + 1583
0x000000010d9ea69a in fast_function (in Python) + 117
0x000000010d9e6af3 in PyEval_EvalFrameEx (in Python) + 26640
0x000000010d9e00f1 in PyEval_EvalCodeEx (in Python) + 1583
0x000000010d9dfabc in PyEval_EvalCode (in Python) + 54
0x000000010da03ea1 in run_mod (in Python) + 53
0x000000010da03f44 in PyRun_FileExFlags (in Python) + 133
0x000000010da03a93 in PyRun_SimpleFileExFlags (in Python) + 698
0x000000010da15445 in Py_Main (in Python) + 3137
0x00007fff857105ad in start (in libdyld.dylib) + 1
```

#6 - 08/04/2016 05:37 PM - Knödlseder Jürgen

Solving the save() method problem was quite trivial: needed to add a test on observation container emptiness:

```
  if (!m_outcube.is_empty() && m_obs.size() > 0) {
    GCTAObservation* obs = dynamic_cast<GCTAObservation*>(m_obs[0]);
```

#7 - 08/04/2016 05:53 PM - Knödlseder Jürgen

The publish() problem was related to a GammaLib problem in the GSkyMap::publish() method. The method segfaulted in fact for an empty sky map.

The GSkyMap::write() method has been modified to return now a pointer to the written (or appended) HDU. In case that nothing is written, the method returns a NULL pointer.

The GSkyMap::publish() checks for a NULL pointer, and simply does nothing in case that a NULL pointer is encountered.

#8 - 08/04/2016 08:35 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 50 to 100

Merged bug fixes into devel.