ctools - Bug #2912

ctbin crashes when reading ebounds fits file for multiple pointings

06/25/2019 02:50 PM - Bernardos Maria Isabel

Status:	Closed	**Start date:**	06/25/2019
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.6.2		

Description

When I tried to use an ebounds file from csebins as energy bins definition in ctbin for a multiple pointing observations, ctbin seems to crash due to paralellization (the crash doesn't happen when setting nthreads=1). I attach the output of doing gdb ctbin for debugging.

History

#1 - 06/25/2019 04:19 PM - Knödlseder Jürgen

I have a suspicion of what happens. According to the debug file, the segmentation fault happens in ctbin::create_cube, which is called in the ctbin::fill_cube method which is inside an OpenMP parallelized loop. In this method, create_ebounds() is called loads the energy boundaries from a FITS file. FITS file access itself is not thread save, which likely leads to the problem.

Modifying the code as follows should solve the issue:

```
GEbounds ctool::create_ebounds(void)
{
   // Allocate energy boundaries
   GEbounds ebounds;

   // Get energy binning algorithm
   std::string ebinalg = (*this)["ebinalg"].string();

   // If energy binning algorithm is of type "FILE" (case sensitive), then
   // read energy boundaries from FITS file ...
   if (ebinalg == "FILE") {

      // Get filename
      GFilename ebinfile = (*this)["ebinfile"].filename();

      // If no extension name was provided then use default extension names
      if (!ebinfile.has_extname()) {

         // Make loading of energy boundaries OpenMP thread save
         #pragma omp critical(ctool_create_ebounds1)
         {

         // Open energy boundary file using the EBOUNDS or ENERGYBINS
         // extension or energy values using the ENERGIES extension.
         // Throw an exception if opening fails.
         GFits file(ebinfile.url());
         if (file.contains(gammalib::extname_ebounds)) {
            file.close();
            ebounds.load(ebinfile);
         }
         else if (file.contains("ENERGYBINS")) {
            file.close();
            ebinfile = ebinfile.url() + "[ENERGYBINS]";
            ebounds.load(ebinfile);
         }
         else if (file.contains(gammalib::extname_energies)) {
            file.close();
            ebounds.set(GEnergies(ebinfile));
         }
         else {
            file.close();
            std::string msg = "No extension with name \""+
                              gammalib::extname_ebounds+"\", "
                              "\"ENERGYBINS\" or \""+
```

```
                        gammalib::extname_energies+"\" found in FITS "
                        "file \""+ebinfile+"\". Please specify a "
                        "valid energy binning file.";
            throw GException::invalid_value(G_CREATE_EBOUNDS, msg);
        }

        } // end: omp critical section

    }

    // ... otherwise load energy boundaries from filename including
    // extension
    else {

        // Make loading of energy boundaries OpenMP thread save
        #pragma omp critical(ctool_create_ebounds2)
        {

        // Open energy boundary file
        GFits file(ebinfile.url());

        // If FITS file does not contain requested extension then throw
        // an exception
        if (!file.contains(ebinfile.extname())) {
            std::string msg = "No extension \""+ebinfile.extname()+"\" "
                        "found in energy binning file \""+
                        ebinfile.url()+"\". Please provide a valid "
                        "extension name.";
            throw GException::invalid_value(G_CREATE_EBOUNDS, msg);
        }

        // Get table extension
        GFitsTable& table = *file.table(ebinfile.extname());

        // If table contains one column then load energies, otherwise load
        // energy boundaries.
        if (table.ncols() == 1) {
            file.close();
            ebounds.set(GEnergies(ebinfile));
        }
        else {
            file.close();
            ebounds.load(ebinfile);
        }

        } // end: omp critical section

    } // endelse: loaded energy boundaries from table extension

} // endif: ebinalg was "FILE"

// ... otherwise use a linear or a logarithmically-spaced energy binning
else {

    // Get task parameters
    double emin     = (*this)["emin"].real();
    double emax     = (*this)["emax"].real();
    int    enumbins = (*this)["enumbins"].integer();

    // Initialise log mode for ebinning
    bool log = true;

    // Check if algorithm is linear
    if (ebinalg == "LIN") {
        log = false;
    }

    // Setup energy bins
    ebounds = GEbounds(enumbins, GEnergy(emin, "TeV"),
                       GEnergy(emax, "TeV"), log);

} // endelse: ebinalg was not "FILE"

// Return energy boundaries
return ebounds;
```

```
}
```

#2 - 06/25/2019 04:54 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Knödlseder Jürgen

- Target version set to 1.6.2

- % Done changed from 0 to 10

I was able to reproduce the problem on a CentOS 7 Linux box:

```
2019-06-25T14:56:06: +==================+
2019-06-25T14:56:06: | Bin observations |
2019-06-25T14:56:06: +==================+
terminate called recursively
terminate called after throwing an instance of 'GException::fits_error'
terminate called recursively
terminate called recursively
terminate called recursively
  what():  *** ERROR in GFits::open(GFilename&, bool&): illegal NAXIS keyword value (status=212).
Aborted (core dumped)
```

#3 - 06/25/2019 05:16 PM - Knödlseder Jürgen

- % Done changed from 10 to 90

Corrected the code as indicated above and no segmentation fault occurs any longer.

#4 - 06/26/2019 10:59 PM - Knödlseder Jürgen

It turned out in the integration tests that no exceptions should be thrown in the critical OpenMP zones, see for example https://stackoverflow.com/questions/13663231/throwing-a-c-exception-inside-an-omp-critical-section.

I therefore reorganised the code as follows:

```
GEbounds ctool::create_ebounds(void)
{
   // Allocate energy boundaries
   GEbounds ebounds;

   // Get energy binning algorithm
   std::string ebinalg = (*this)["ebinalg"].string();

   // If energy binning algorithm is of type "FILE" (case sensitive), then
```

```
// read energy boundaries from FITS file ...
if (ebinalg == "FILE") {

    // Get filename
    GFilename ebinfile = (*this)["ebinfile"].filename();

    // Initialise exception message string
    std::string msg;

    // Make loading of energy boundaries OpenMP thread save. The critical
    // region is put into an exception block to prevent throwing exceptions
    // within the critical block
    #pragma omp critical(ctool_create_ebounds)
    {
        try {

            // If no extension name was provided then use default extension
            // names
            if (!ebinfile.has_extname()) {

                // Open energy boundary file using the EBOUNDS or ENERGYBINS
                // extension or energy values using the ENERGIES extension.
                // If opening fails then set exception message. The
                // exception is thrown later outside the critical OpenMP
                // block.
                GFits file(ebinfile.url());
                if (file.contains(gammalib::extname_ebounds)) {
                    file.close();
                    ebounds.load(ebinfile);
                }
                else if (file.contains("ENERGYBINS")) {
                    file.close();
                    ebinfile = ebinfile.url() + "[ENERGYBINS]";
                    ebounds.load(ebinfile);
                }
                else if (file.contains(gammalib::extname_energies)) {
                    file.close();
                    ebounds.set(GEnergies(ebinfile));
                }
                else {
                    file.close();
                    msg = "No extension with name \""+
                          gammalib::extname_ebounds+"\", "
                          "\"ENERGYBINS\" or \""+
                          gammalib::extname_energies+"\" found in FITS "
                          "file \""+ebinfile+"\". Please specify a "
                          "valid energy binning file.";
                }

            } // endif: no extension name specified

            // ... otherwise load energy boundaries from filename including
            // extension
            else {

                // Open energy boundary file
                GFits file(ebinfile.url());

                // If FITS file does not contain requested extension then
                // set exception message. The exception is thrown later
                // outside the critical OpenMP zone
                if (!file.contains(ebinfile.extname())) {
                    msg = "No extension \""+ebinfile.extname()+"\" "
                          "found in energy binning file \""+
                          ebinfile.url()+"\". Please provide a valid "
                          "extension name.";
                }

                // ... otherwise load energy boundaries
                else {

                    // Get table extension
                    GFitsTable& table = *file.table(ebinfile.extname());

                    // If table contains one column then load energies,
```

```
                    // otherwise load energy boundaries.
                    if (table.ncols() == 1) {
                        file.close();
                        ebounds.set(GEnergies(ebinfile));
                    }
                    else {
                        file.close();
                        ebounds.load(ebinfile);
                    }

                } // endelse: extension name was present in FITS file

            } // endelse: loaded energy boundaries from table extension

        } // endtry

        // Catch any exceptions that occured in OpenMP critical block
        // and recover the error message
        catch (const std::exception& e) {
            msg = e.what();
        }

    } // end critical OpenMP block

    // If we have an exception message then throw an exception now. We
    // have to do this since we should not throw exceptions inside OpenMP
    // critical blocks
    if (!msg.empty()) {
        throw GException::invalid_value(G_CREATE_EBOUNDS, msg);
    }

  } // endif: ebinalg was "FILE"

  // ... otherwise use a linear or a logarithmically-spaced energy binning
  else {

    // Get task parameters
    double emin     = (*this)["emin"].real();
    double emax     = (*this)["emax"].real();
    int    enumbins = (*this)["enumbins"].integer();

    // Initialise log mode for ebinning
    bool log = true;

    // Check if algorithm is linear
    if (ebinalg == "LIN") {
        log = false;
    }

    // Setup energy bins
    ebounds = GEbounds(enumbins, GEnergy(emin, "TeV"),
                       GEnergy(emax, "TeV"), log);

  } // endelse: ebinalg was not "FILE"

  // Return energy boundaries
  return ebounds;
}
```

#5 - 07/01/2019 09:24 AM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 90 to 100

Merged into bugfix-1.6.2 and also the devel branch.

Files

debug_ctbin.log	6.2 KB	06/25/2019	Bernardos Maria Isabel