Subject: Usage of exceptions in CINT
Posted by Philipp Bielefeldt on Wed, 17 Feb 2016 11:41:54 GMT
View Forum Message <> Reply to Message

Hi forum!
I am working on a CINT macro with FopiRoot, but I seem unable to get the exception handling done correctly. I set up a Kalman fitter kFitter and create a track that I want to process. As long as the fitter doesn't throw and exception, it seems to work -- however, in case the GFKalman fails, the following happens:

```
std::cout << "Entries in Track: " << NHitsInRecoTrack << std::endl;
try{
   kFitter->processTrack(recoTrack);
}
catch(const GFException& e){
   std::cerr << e.what() << std::endl;
   std::cout << "[WARN    ] Exception in track procession, next track" << std::endl;
   continue;
}
std::cout << "[INFO    ] Processed track!" << std::endl;
```

This is my code -- I put an cli output just before and after the try/catch-block.
The result (in a case an exception is thrown) is:

```
Entries in Track: 7
GFException thrown with excString:
RKTrackRep::RKutta ==> momentum too low: 1.08513 MeV
in line: 1215 in file:
/home/bielefeldt/g2fopiroot/GenfitTools/trackrep/RKTrackRep/RKTrackRep.cxx
with fatal flag 0
[INFO    ] Processed track!
```

Apparently, the exception is printed. But this also happens if I comment out the e.what-line; The "[WARN   ]..." and the "continue" seem to be ignored. However, even if I do not ask for the exception to be printed, it will be written to console.

The CINT documentation is a bit vague, but states
Quote:
Exception handling is implemented but with limitations. Object destruction may not be done properly in complex case. It is recommended not to use Exception Handling in interpreted code.

which is why I use interpreted code (loading with .L XYZ.C++).

I am quite puzzled -- maybe I am doing s.th. truly awful, but I do not understand how I am supposed to work properly with exception handling on the fitter. If anyone here can give me a little more understanding (or any useful reference), I'd be very thankful!

Cheers!
Philipp

Subject: Re: Usage of exceptions in CINT
Posted by Ralf Kliemt on Tue, 23 Feb 2016 08:31:11 GMT
View Forum Message <> Reply to Message

Hi,

Are you sure a const GFException& is thrown outside of GenFit? Maybe it's a standar exception being thrown or everything was handld inside already and the fitter returned "normally"?

Subject: Re: Usage of exceptions in CINT
Posted by Philipp Bielefeldt on Wed, 24 Feb 2016 17:46:13 GMT
View Forum Message <> Reply to Message

Maybe I do not get the point in your question accurately... But the object thrown is a GFException, and the referenced line RKTrackRep.cxx:1215 reads

```
GFException exc(sstream.str(),__LINE__,__FILE__);
exc.setFatal();
throw exc;
```

It is a bit unclear to me what you mean with "outside of GenFit". I am still working with CINT linked against Genfit... so it is not really "outside", but maybe I am misled here.
I cannot see how this could be a default/empty/... exception. As far as I can see some thing bewildering happens with the thrown exception. And by the way: I could reproduce the same weird behaviour in a "neatly compiled" class, where I wrapped up the ->processTrack like so:

```
bool GFExecution::cwProcessTrack(GFAbsFitter* aFitter, GFTrack* aTrack){
   fStatus = true;
   std::cout << "GFExecution::cwProcessTrack... Status is " << fStatus << std::endl;

   try{
      aFitter->processTrack(aTrack);
   }catch(const GFException& e){
      fStatus = false;
      std::cout << "[WARN    ] Exception in track procession -- ProcessTrack set to false" << std::endl;
      std::cerr << e.what() << std::endl;
   }

   std::cout << "    return status is " << fStatus << std::endl;
   return fStatus;
}
```

but again, in case an exception is thrown, the exception's content appears on the cli, whilst the catch block itself seems to be not executed, it looks like so:

```
GFExecution::cwProcessTrack... Status is 1
GFException thrown with excString:
RKTrackRep::RKutta ==> momentum too low: 3.53823 MeV
in line: 1215 in file:
/home/bielefeldt/g2fopiroot/GenfitTools/trackrep/RKTrackRep/RKTrackRep.cxx
with fatal flag 0
   return status is 1
```

Most likely, I am doing some thing very incorrect in the way I implement exception handling, but I just cannot find my flaw.

Subject: Re: Usage of exceptions in CINT
Posted by Ralf Kliemt on Wed, 24 Feb 2016 18:19:27 GMT
View Forum Message <> Reply to Message

Hi Pilipp.

I expect GenFit to catch its exceptions by itself. You see the printout of a catch inside and GenFit proceeds in a conditioned way (i.e. exiting properly the fitting routine).
You may put a cout at the end inside your try() to see that you don't get something thrown at you without expecting it.

To filter bad fits there must be a flag or fit status somewhere.

Cheers!
 Ralf

Subject: Re: Usage of exceptions in CINT
Posted by Elisabetta Prencipe (2) on Fri, 26 Feb 2016 13:52:06 GMT
View Forum Message <> Reply to Message

Hello Philipp,
 if you are using genfit(rev-400) you can check the convergence of your fit with FitStatus(), that gives access to infrmation like chi2, nDof,.... Or simply check the flag "GetFlag()". The easiest way is to run sim- digi- reco- pid- macro, then open the pid_complete.root file and check the following:

```
  cbmsim->Draw("SttMvdGemGenTrack.GetFlag()")
```

In GenfitTools/adapters the genfit tracks are transformed in PndTracks. The flag is set to -1 there in, if the fit fails after a number of iterations that you can set in your reco-macro .

In genfit2 (rev > 1765) indeed you can find:

Track::getFitStatus(const AbsTrackRep* rep = NULL)

You can check these kind of flags to decide whether to keep/count a Track or not. Concerning the Exceptions: the fitter should catch all Exceptions, and if the fit fails, set the FitStatus() accordingly. So processTrack() should not throw any Exceptions at all.

I hope it helps. I am discussing with the convenors the possibility to give a talk on the differences genfit vs genfit2 the next week, in Bochum.

cheers,
Elisabetta