Subject: compilation of macros
Posted by Bertram Kopf on Tue, 01 Dec 2009 20:40:46 GMT
View Forum Message <> Reply to Message

Dear all,
as decided during the last PandaRoot meeting, I am currently developing tools to automatize the link/run procedure for all macros. Very first rudimental tools are already existing on my local version. As a starting point I used the properly running emc macro "sim_emc.C" where a slight modification was needed by e.g. including the relevant header files. The first trial to compile this modified macro failed, since the forward declaration of a class, namely "class PndGeoDrc" in "PndDrc.h", was missing. After fixing this bug, the macro has been compiled successfully, but unfortunately with a lot of warnings in the log output:

Toggle Spoiler

Processing runMacros.C...
Warning in <TClassTable::Add>: class PndSttTrack already in TClassTable
Warning in <TClassTable::Add>: class PndMvdPixel already in TClassTable
Warning in <TClassTable::Add>: class PndMvdStrip already in TClassTable
Warning in <TClassTable::Add>: class PndMvdMCPoint already in TClassTable
Warning in <TClassTable::Add>: class PndMvdDigi already in TClassTable
Warning in <TClassTable::Add>: class PndMvdDigiPixel already in TClassTable
Warning in <TClassTable::Add>: class PndMvdApvHit already in TClassTable
Warning in <TClassTable::Add>: class PndMvdPidCand already in TClassTable
Info in <TUnixSystem::ACLiC>: creating shared library
/home/bertram/PndRoot/091123/trunk/macro/emc/./AbsArgTest_C.so
Info in <TUnixSystem::ACLiC>: creating shared library
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C.so
In file included from /home/bertram/PndRoot/091123/trunk/base/FairRunSim.h:9,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:8,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TObject.h:125: warning: ‘virtual const char*
TObject::GetName() const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairParticle.h:46: warning: by ‘const TString&
FairParticle::GetName()’
/FairRootExt/july09/tools/root/include/TObject.h:144: warning: ‘virtual void
TObject::Print(const Option_t*) const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairParticle.h:37: warning: by ‘void
FairParticle::Print() const’
In file included from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:9,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TNamed.h:59: warning: ‘virtual void
TNamed::Print(const Option_t*) const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairModule.h:34: warning: by ‘virtual void
FairModule::Print() const’
/home/bertram/PndRoot/091123/trunk/base/FairModule.h:55: warning: unused parameter

‘vname’
In file included from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:10,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/base/FairDetector.h:60: warning: unused parameter
‘cl1’
/home/bertram/PndRoot/091123/trunk/base/FairDetector.h:60: warning: unused parameter
‘cl2’
/home/bertram/PndRoot/091123/trunk/base/FairDetector.h:60: warning: unused parameter
‘offset’
In file included from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:11,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/base/FairModule.h:53: warning: ‘virtual void
FairModule::ExpandNode(TGeoNode*)’ was hidden
/home/bertram/PndRoot/091123/trunk/emc/EmcMC/PndEmc.h:116: warning: by ‘void
PndEmc::ExpandNode(TGeoVolume*, TGeoVolume*)’
In file included from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndMdt.h:14,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:12,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TObject.h:152: warning: ‘virtual Int_t
TObject::Write(const char*, Int_t, Int_t)’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairRootManager.h:69: warning: by ‘void
FairRootManager::Write()’
/FairRootExt/july09/tools/root/include/TObject.h:153: warning: ‘virtual Int_t
TObject::Write(const char*, Int_t, Int_t) const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairRootManager.h:69: warning: by ‘void
FairRootManager::Write()’
In file included from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndMdt.h:16,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:12,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TNamed.h:51: warning: ‘virtual const char*
TNamed::GetName() const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairVolume.h:29: warning: by ‘const char*
FairVolume::GetName()’
In file included from /home/bertram/PndRoot/091123/trunk/parbase/FairParGenericSet.h:5,
from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndGeoMdtPar.h:5,
from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndMdt.h:18,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:12,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from

/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/parbase/FairParSet.h:22: warning: unused parameter
‘io’
In file included from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndGeoMdtPar.h:5,
from /home/bertram/PndRoot/091123/trunk/mdt/MdtMC/PndMdt.h:18,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:12,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/parbase/FairParSet.h:23: warning: ‘virtual Int_t
FairParSet::write()’ was hidden
/home/bertram/PndRoot/091123/trunk/parbase/FairParGenericSet.h:20: warning: by
‘virtual Int_t FairParGenericSet::write(FairParIo*)’
/home/bertram/PndRoot/091123/trunk/parbase/FairParSet.h:21: warning: ‘virtual Bool_t
FairParSet::init()’ was hidden
/home/bertram/PndRoot/091123/trunk/parbase/FairParGenericSet.h:19: warning: by
‘virtual Bool_t FairParGenericSet::init(FairParIo*)’
In file included from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:13,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/base/FairDetector.h:60: warning: ‘virtual void
FairDetector::CopyClones(TClonesArray*, TClonesArray*, Int_t)’ was hidden
/home/bertram/PndRoot/091123/trunk/drc/PndDrc.h:102: warning: by ‘virtual void
PndDrc::CopyClones(TClonesArray*, TClonesArray*, TClonesArray*, TClonesArray*,
Int_t)’
In file included from /home/bertram/PndRoot/091123/trunk/field/PndMultiField.h:18,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:16,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TNamed.h:59: warning: ‘virtual void
TNamed::Print(const Option_t*) const’ was hidden
/home/bertram/PndRoot/091123/trunk/base/FairField.h:123: warning: by ‘virtual void
FairField::Print()’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:97: warning: unused parameter
‘x’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:97: warning: unused parameter
‘y’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:97: warning: unused parameter
‘z’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:103: warning: unused parameter
‘x’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:103: warning: unused parameter
‘y’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:103: warning: unused parameter
‘z’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:109: warning: unused parameter
‘x’

/home/bertram/PndRoot/091123/trunk/base/FairField.h:109: warning: unused parameter ‘y’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:109: warning: unused parameter ‘z’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:125: warning: unused parameter ‘point’
/home/bertram/PndRoot/091123/trunk/base/FairField.h:125: warning: unused parameter ‘bField’
In file included from /home/bertram/PndRoot/091123/trunk/emc/EmcData/PndEmcHit.h:17,
from /home/bertram/PndRoot/091123/trunk/emc/EmcDigi/PndEmcHitProducer.h:14,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:20,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/base/FairHit.h:65: warning: unused parameter ‘opt’
In file included from /home/bertram/PndRoot/091123/trunk/emc/EmcDigi/PndEmcHitProducer.h:18,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:20,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/FairRootExt/july09/tools/root/include/TObject.h:144: warning: ‘virtual void TObject::Print(const Option_t*) const’ was hidden
/home/bertram/PndRoot/091123/trunk/emc/EmcTools/PndEmcStructure.h:39: warning: by ‘void PndEmcStructure::Print(std::string, Int_t) const’
In file included from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. h:34,
from /home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new_C_ACLiC_dict. cxx:17:
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C: In function ‘void sim_emc_new(Int_t, Float_t)’:
/home/bertram/PndRoot/091123/trunk/macro/emc/./sim_emc_new.C:76: warning: unused variable ‘Pipe’

This example shows that the pre-compilation of the macros could result in a helpful diagnostic tool to improve the software, even for macros which seem to work properly.

Best regards,
Bertram.

Subject: Re: compilation of macros
Posted by Mohammad Al-Turany on Tue, 01 Dec 2009 22:01:12 GMT
View Forum Message <> Reply to Message

Hi Bertram,

I added the missing declaration to the PndDrc.h. For me it is a mystery, because the PndDrc class is always compiled (libPndDrc.so) and how could this happened with all compilers and systems? even the CC and ICC did not detect it.

Anyway it is corrected now.

regards

Mohammad

Subject: Re: compilation of macros
Posted by Mohammad Al-Turany on Wed, 02 Dec 2009 09:03:13 GMT
View Forum Message <> Reply to Message

Hi,

Now it is not a mystery any more, this has worked because of the wrong order of includes in the PndDrc.cxx. i.e:

```
#include "PndDrcSurfPolyFlat.h"
#include "PndDrcOptReflSilver.h"
#include "PndDrcOptMatLithotecQ0.h"
#include "PndDrcOptDevSys.h"
#include "PndDrcOptVol.h"
#include "PndDrcOptDevManager.h"
#include "PndGeoDrc.h"
#include "PndDrcPDPoint.h"
#include "PndDrcBarPoint.h"
#include "PndGeoDrcPar.h"
#include "PndDetectorList.h"
#include "PndDrc.h"

....
```

so as you can see the PndGeoDrc.h is included before the PndDrc.h, that is why the compiler always find it.

regards

Mohammad

Subject: Re: compilation of macros
Posted by Florian Uhlig on Wed, 02 Dec 2009 09:50:14 GMT
View Forum Message <> Reply to Message

Hi Bertram

If the order of the header files is not correct the you can end up in realy funny situations. This is exactely what you saw here.
The header file of the PndDrc class was not correct but the error was hidden by the incorrect order in the source file.

The rule of thumb is to order the header files from local to global. That means start with the header file for the class, then
the header files for the module, then the header files for the project, the all other header files and in the end the system header files.

Ciao

Florian

Subject: Re: compilation of macros
Posted by Bertram Kopf on Wed, 02 Dec 2009 16:19:30 GMT
View Forum Message <> Reply to Message

Hi all,
meanwhile, I tried to compile the second emc macro "digi_emc.C". It didn't compile since "TString parFile" is declared twice within the macro. This is definitely a bug, even though the original macro w/o the compilation procedure seems to run properly. Can somebody tell me, why the non-compiled macro is running at all. No c++ compiler would accept such kind of code.
After getting rid of the second declaration everything works fine.

Cheers,
Bertram.

Subject: Re: compilation of macros
Posted by StefanoSpataro on Wed, 02 Dec 2009 16:25:09 GMT
View Forum Message <> Reply to Message

I can tell you,
the "macros" are just macros, they are not compiled at all.
ROOT CINT is an interpreter, not a compiler, and sometimes it "fixes" eventual problems occurring in the macros. This is one case.
Another case is that there is no destruction at the end of the context, then in a macro you can define a variable inside brackets and access to it in the root line, or it defines not initialized variables.

Of course, the compiled code inside shared libraries is compiled then all the mentioned errors crash the compilation, therefore cannot occur.

Subject: Re: compilation of macros

Posted by Bertram Kopf on Wed, 02 Dec 2009 17:44:52 GMT
View Forum Message <> Reply to Message

Hi Stefano,
you are completely right, ROOT CINT is an interpreter. I just wanted to point out that ROOT CINT seems to have no protection mechanism against such kind of code.

Cheers,
Bertram.

Subject: Re: compilation of macros
Posted by Matthias Steinke on Wed, 02 Dec 2009 18:22:35 GMT
View Forum Message <> Reply to Message

Hi everybody,

did I understood correctly, that the order of #include statements has some effect?? This must be a joke. If not, what about cleaning up all header files?

Matthias

Subject: Re: compilation of macros
Posted by Mohammad Al-Turany on Wed, 02 Dec 2009 18:57:25 GMT
View Forum Message <> Reply to Message

Hi,

The order of includes should not make any difference. The Joke is if one order the includes from local to global this will help identifying such cases. i.e. one get an error if the header depends on a standard (or local) header but doesn't itself #include that header.

And that was the case here. If the PndDrc.h was included first then the compiler will find the error, but the facts that

1. the header was never used directly (tell Bertram tried to compile the macro!)
2. the PndDrcGeo.h was included in the cxx file before the PndDrc.h

make this situation.

So it is a good Joke.

Quote:what about cleaning up all header files?

This would be great, but we cannot do this all the time ourselves, the people writing code should also take care of there own code, and following such jokes as the one here will help all to find such situations more easily.

So one can consider this issue of include order as a convention or an advice (or a Joke!) and the fact that C++ does not forbid this or even care about it allow us to ask for it.

regards

Mohammad

Subject: Re: compilation of macros
Posted by Elwin Dijck on Wed, 02 Dec 2009 22:24:30 GMT
View Forum Message <> Reply to Message

I also experimented with compiling macros and actually used g++ instead of ACLiC for compilation and then linked with the PandaRoot libraries. The linking part gave me several errors about undefined symbols, because the PandaRoot libraries contain (public) functions that are declared but not implemented. I guess it should be possible to ignore the errors as none of these functions are ever called, but in any case it might be better to remove their declarations from the class interfaces. This won't break any code as calling these function would result in crashes anyway.

I compiled a list of functions that are declared, but don't seem to be implemented. It might be good to comment-out the declarations for functions that are supposed to be implemented at some point or otherwise just remove them (or make them private).

List

dch/PndDchDrifter.h:59 void PndDchDrifter::Initialize()
emc/EmcDigi/PndEmcApdHitProducer.h:42 PndEmcApdHit *PndEmcApdHitProducer::AddHit(Int_t, Int_t, Float_t, Float_t)
emc/EmcDigi/PndEmcApdHitProducer.h:46 PndEmcApdHit *PndEmcApdHitProducer::AddHit(Int_t, Int_t, Float_t, Float_t, vector<PndEmcApdPoint *>)
emc/EmcDigi/PndEmcHitProducer.h:48 PndEmcHit *PndEmcHitProducer::AddHit(Int_t, Int_t, Float_t, Float_t, vector<PndEmcPoint *>)
emc/EmcData/PndEmcWaveform.h:98 int PndEmcWaveform::getWaveformLength() const
emc/EmcData/PndEmcCluster.h:77 Int_t PndEmcCluster::thetaIndexInt() const
emc/EmcData/PndEmcCluster.h:80 Int_t PndEmcCluster::phiIndexInt() const
gem/PndGemSensor.h:126 Bool_t PndGemSensor::ActivateChannels(Int_t, Double_t, Double_t)
gem/PndGemSensor.h:137 Int_t PndGemSensor::Intersect(Int_t, Int_t, vector<Double_t> &, vector<Double_t> &)
gem/PndGemSensor.h:145 Int_t PndGemSensor::IntersectClusters(Double_t, Double_t, Double_t &, Double_t &, Double_t &)
gem/PndGemSensor.h:154 Int_t PndGemSensor::PointIndex(Int_t, Int_t)
hyp/hypDigi/PndHypCalcStrip.h:80 ostream &PndHypCalcStrip::operator<<(ostream &)
lhetrack/PndLheTrackCuts.h:60 Double_t PndLheTrackCuts::GetDelY(Int_t) (implementation trivial)
lhetrack/PndLheTrackCuts.h:81 Double_t PndLheTrackCuts::TrackHitAlpha(PndLheCMCandidate *, PndLheCMPoint *, Bool_t)

lhetrack/PndLheTrackCuts.h:82 Double_t PndLheTrackCuts::TrackHitCircleDist(PndLheCMCandidate *, PndLheCMPoint *, Bool_t)
lhetrack/PndLheSegments.h:56 Int_t PndLheSegments::GetRadiusSegm(const PndLheCMPoint *)
lhetrack/PndLheSegments.h:57 Int_t PndLheSegments::GetRadiusSegm(Int_t)
lhetrack/PndLheSegments.h:64 Int_t PndLheSegments::GetStation(Int_t)
lhetrack/PndLheHitsMaker.h:105 void PndLheHitsMaker::GetStripPoints()
lumi/LumiDigi/PndLumiStripHitProducer.h:46 void PndLumiStripHitProducer::Print() const
mvd/MvdDigi/PndMvdCalcStrip.h:106 ostream &PndMvdCalcStrip::operator<<(ostream &)
mvd/MvdTracking/PndMvdTpcRiemannCorrelatorTask.h:32 void PndMvdTpcRiemannCorrelatorTask::PrintResult()
mvd/MvdTracking/PndMvdTPCRiemannTrackFinderTaskEff.h:32 void PndMvdTPCRiemannTrackFinderTaskEff::PrintResult()
mvd/MvdTracking/PndMvdRiemannTrackFinderTask.h:32 void PndMvdRiemannTrackFinderTask::PrintResult()
mvd/MvdTracking/PndTpcClustPlusRTFTask.h:26 void PndTpcClustPlusRTFTask::PrintResult()
mvd/MvdTracking/PndMvdTrackFinderAnaTask.h:30 void PndMvdTrackFinderAnaTask::PrintResult()
mvd/MvdTracking/PndMvdTPCRiemannTrackFinderTask.h:39 void PndMvdTPCRiemannTrackFinderTask::PrintResult()
mvd/MvdTracking/PndMvdRiemannVertexFinderTask.h:28 void PndMvdRiemannVertexFinderTask::PrintResult()
mvd/MvdTracking/PndMvdRiemannTrackFinderTaskCutPar.h:35 void PndMvdRiemannTrackFinderTaskCutPar::PrintResult()
mvd/MvdTracking/PndMvdRiemannTrackFinderTaskEff.h:32 void PndMvdRiemannTrackFinderTaskEff::PrintResult()
mvd/MvdTracking/PndTpcClustPlusRTFTaskCutPar.h:36 void PndTpcClustPlusRTFTaskCutPar::PrintResult()
mvd/MvdTracking/PndMvdTPCRiemannTrackFinderTaskCutPar.h:37 void PndMvdTPCRiemannTrackFinderTaskCutPar::PrintResult()
pnddata/MvdData/PndMvdDigiStrip.h:32 PndMvdDigiStrip::PndMvdDigiStrip(Int_t, Int_t, TString, Int_t, Int_t, Double_t)
pnddata/TrackData/PndTrackID.h:38 void PndTrackID::Sort()
pnddata/SttData/PndSttGeomPoint.h:57 void PndSttGeomPoint::Transform()
pnddata/SttData/PndSttPoint.h:52 void PndSttPoint::SetMomentumtot(TVector3)
pnddata/MdtData/PndMdtHit.h:25 void PndMdtHit::Clear()
PndTools/AnalysisTools/Fitter/PndVtxFitter.h:107 void PndVtxFitter::SetBeamProfile(const TMatrixDSym &)
PndTools/AnalysisTools/Fitter/PndVtxFitter.h:108 void PndVtxFitter::SetVertexProfile(const TMatrixDSym &)
PndTools/AnalysisTools/Fitter/PndVtxFitter.h:110 TMatrixDSym PndVtxFitter::GetVertexProfile() const
PndTools/AnalysisTools/Fitter/PndVtxFitterParticle.h:91 PndVtxFitterParticle &PndVtxFitterParticle::operator=(const PndVtxFitterParticle &) # should propably be private
stt/PndSttSingleStraw.h:121 Int_t PndSttSingleStraw::StrawTot()
stt/PndSttTrackFinderIdeal.h:42 void PndSttTrackFinderIdeal::ZoomTrack(Double_t &, Double_t &, Double_t &, PndSttTrack *)
tpc/DebugLogger.h:56 unsigned int DebugLogger::addOutFile(string)
tpc/tpcreco/test/PndTpcRecoTester.h:50 void PndTpcRecoTester::testRiemannTrack()
tpc/PndTpcLaserFitTask.h:57 void PndTpcLaserFitTask::setOpeningAngle(double)
trackbase/FairTrackParP.h:67 void FairTrackParP::SetTrackPar(Double_t, Double_t,

Double_t, Double_t, Double_t, Double_t, Int_t, Double_t[15])
trackbase/FairTrackParP.h:70 void FairTrackParP::SetTrackPar(Double_t, Double_t, Double_t, Double_t, Double_t, Double_t[15])

Additionally, the PndMvdGeo class is referenced in the code, but the source code file PndMvdGeo.cxx is excluded from the build by default (mvd/MvdLinkDef.h:19 and mvd/CMakeLists.txt:62), which also results in undefined references when externally linking.

Subject: Re: compilation of macros
Posted by Bertram Kopf on Wed, 02 Dec 2009 23:40:32 GMT
View Forum Message <> Reply to Message

Hi Elwin,

Elwin Dijck wrote on Wed, 02 December 2009 23:24I also experimented with compiling macros and actually used g++ instead of ACLiC for compilation and then linked with the PandaRoot libraries. The linking part gave me several errors about undefined symbols, because the PandaRoot libraries contain (public) functions that are declared but not implemented. I guess it should be possible to ignore the errors as none of these functions are ever called, but in any case it might be better to remove their declarations from the class interfaces. This won't break any code as calling these function would result in crashes anyway.

These errors should not be ignored. In any case, one has to fix it! Your example shows that it is might be better to link with g++ instead of ACLiC.
Therefore I would like to ask everybody to think about the idea to get rid of the existing macros and create binaries with g++ instead. In order to steer the applications, the application framework can provide an additional interpreter with the purpose to enable/disable tasks, communicate with the tasks, etc. I know, it is just an idea and it would require lots of changes. Anyhow, it would be great to think about such an idea.

Cheers,
Bertram.

Subject: Re: compilation of macros
Posted by Florian Uhlig on Thu, 03 Dec 2009 07:52:56 GMT
View Forum Message <> Reply to Message

Hi Bertram

Quote:
meanwhile, I tried to compile the second emc macro "digi_emc.C". It didn't compile since "TString parFile" is declared twice within the macro. This is definitely a bug, even though the original macro w/o the compilation procedure seems to run properly. Can somebody tell me, why the non-compiled macro is running at all. No c++ compiler would accept such kind of

code.
After getting rid of the second declaration everything works fine.

This is due to the fact that ROOTCINT is an interpreter and the
developers made it as foolproofe as possible. If this was a good choice one can debate, but most users of ROOT are physicist who don't care about coding conventions, ordering of includes, catching any error and so on. They are interested in results and this as fast and easy as possible which is okay and understandable but which contradicts with good and clean coding. The second and in my opinion much more important point is that at most universities there are no programing lessons at all and i think with the new bachalor/master studies things even get worse.

Ciao

Florian

Subject: Re: compilation of macros
Posted by Johan Messchendorp on Thu, 03 Dec 2009 08:34:16 GMT
View Forum Message <> Reply to Message

Dear all,

Lets not get too excited here and start a war about the usage of ROOTCINT! We really have other priorities to focus on. First of all, the reason why we decided to compile the macros is to clean up the existing code and find potential problems. And, indeed it proofs to be useful, hurray!. So, many thanks to Bertram and Elwin!

For the longer term, I would propose is to have as - a QA check! - the "all-in-one-macros" compiled and linked as well as binary. The outcome of such a compilation, we can place on the dashboard, so that the developers can take a look at it and use it to clean up their code. For the rest, I am very very very hesitative to start forgetting about CINT. I like it as a steering facility (flexible, simple, ...), like many many many others... And that fact also has an important value, do not underestimate it!

Greetings,

Johan.

Subject: Re: compilation of macros
Posted by Florian Uhlig on Thu, 03 Dec 2009 08:48:20 GMT
View Forum Message <> Reply to Message

Hi Mathias

Quote:did I understood correctly, that the order of #include statements has some effect?? This must be a joke. If not, what about cleaning up all header files?

It is not a joke. This is real life and a well known problem.

If all the header files would be both self sufficient and idempotent then the order would not matter. As you can see with
this problem if they are not you can have problems.
I add below a part from Bruce Eckels book "Thinking in C++, Volume 1" which hopefully makes the point more clear.

Quote:
Order of header inclusion

Headers are included from "the most specific to the most general." That is, any header files in the local directory are included first, then any of my own "tool" headers such as require.h or purge.h, then any third-party library headers, then the standard C++ library headers, and finally the C library headers.

The justification for this comes from John Lakos in Large-Scale C++ Software Design (Addison-Wesley, 1996):

Latent usage errors can be avoided by ensuring that the .h file of a component parses by itself -- without externally-provided declarations or definitions... Including the .h file as the very first line of the .c file ensures that no critical piece of information intrinsic to the physical interface of the component is missing from the .h file (or, if there is, that you will find out about it as soon as you try to compile the .c file).

If the order of header inclusion goes "from most specific to most general," then it's more likely that if your header doesn't parse by itself, you'll find out about it sooner and prevent annoyances down the road.

By the way the PndDrc.h file was the first header included in PndDrc.cxx until 22.07.09. At that point the order was changed, probaly to overcome the problem in the header file which came in at the same time.

To clean all header files would be a good idea but i fear that it will not help. We did this some time ago and now it is again a mess.

Ciao

Florian

Subject: Re: compilation of macros
Posted by StefanoSpataro on Thu, 03 Dec 2009 08:54:06 GMT
View Forum Message <> Reply to Message

Question:

how this can be considered as "error", considering that the results are exactly the same?

Maybe it is just a bit "un-elegant", or am I missing something important?

Subject: Re: compilation of macros
Posted by Florian Uhlig on Thu, 03 Dec 2009 10:23:00 GMT
View Forum Message <> Reply to Message

Hi Bertram

Quote:
These errors should not be ignored. In any case, one has to fix it! Your example shows that it is might be better to link with g++ instead of ACLiC.
Therefore I would like to ask everybody to think about the idea to get rid of the existing macros and create binaries with g++ instead. In order to steer the applications, the application framework can provide an additional interpreter with the purpose to enable/disable tasks, communicate with the tasks, etc. I know, it is just an idea and it would require lots of changes. Anyhow, it would be great to think about such an idea.

I think it is good to look for potential problems, so it would also make sense to take a look at all the warnings which show up during compilation. They already show some of the problems.

The macros are meant for steering the execution of the compiled code and nothing else. If you want to compile them this is fine to find some errors which wer hidden up to now by the behaviour of rootcint.
But i don't understand why you want to get rid of the interpreter and invent a new one. as you said the requires alot of changes and i don't know who should do this if the people not even clean their own code after the did some changes or care about warnings in their part of code.

Ciao

Florian

Subject: Re: compilation of macros
Posted by Bertram Kopf on Thu, 03 Dec 2009 13:48:34 GMT
View Forum Message <> Reply to Message

Hi Florian,

Quote:
I think it is good to look for potential problems, so it would also make sense to take a look at all the warnings which show up during compilation. They already show some of the problems.

The macros are meant for steering the execution of the compiled code and nothing else. If you want to compile them this is fine to find some errors which wer hidden up to now by the behaviour of rootcint.
But i don't understand why you want to get rid of the interpreter and invent a new one. as you said the requires alot of changes and i don't know who should do this if the people not

even clean their own code after the did some changes or care about warnings in their part of code.

as you already wrote, the creation of binaries or the compilation of the macros would result in an additional diagnostic tool for the QA of the code.
The idea of introducing a new interpreter is as follows:
With ROOTCINT I don't see the possibility to interact with the framework in a sufficient way. Apart from the general steering of the applications, it would be helpful if the interpreter is also able to interact in a comfortable way with the tasks, the sub-tasks of the tasks and -if needed- also with the objects which are treated within the (sub)tasks. Examples are the setting of parameters or parameter sets for each module, the enabling/disabling or the cloning of modules etc at runtime. For such purposes I see some limitations by using ROOTCINT. Am I wrong?

Best regards,
Bertram.

Subject: Re: compilation of macros
Posted by Florian Uhlig on Fri, 04 Dec 2009 13:40:42 GMT
View Forum Message <> Reply to Message

Hi Bertram

Quote:The idea of introducing a new interpreter is as follows:
With ROOTCINT I don't see the possibility to interact with the framework in a sufficient way. Apart from the general steering of the applications, it would be helpful if the interpreter is also able to interact in a comfortable way with the tasks, the sub-tasks of the tasks and -if needed- also with the objects which are treated within the (sub)tasks. Examples are the setting of parameters or parameter sets for each module, the enabling/disabling or the cloning of modules etc at runtime. For such purposes I see some limitations by using ROOTCINT. Am I wrong?

I don't understand what you want? Could you explain what you want to do? What do you mean with cloning of modules at runtime?

Ciao

Florian

Subject: Re: compilation of macros
Posted by Elwin Dijck on Fri, 04 Dec 2009 14:29:43 GMT
View Forum Message <> Reply to Message

The declarations of functions that were not implemented are now commented-out, except for the following two in FairTrackParP. Can someone please look at those?

trackbase/FairTrackParP.h:67 void FairTrackParP::SetTrackPar(Double_t, Double_t, Double_t,

Double_t, Double_t, Double_t, Int_t, Double_t[15])
trackbase/FairTrackParP.h:70 void FairTrackParP::SetTrackPar(Double_t, Double_t, Double_t, Double_t, Double_t, Double_t[15])

-Elwin

Subject: Re: compilation of macros
Posted by Mohammad Al-Turany on Fri, 04 Dec 2009 14:38:29 GMT
View Forum Message <> Reply to Message

HI Elwin,

it is in SVN now.

Mohammad

Subject: Re: compilation of macros
Posted by Bertram Kopf on Fri, 04 Dec 2009 15:32:01 GMT
View Forum Message <> Reply to Message

Hi Florian,
Quote:
I don't understand what you want? Could you explain what you want to do? What do you mean with cloning of modules at runtime?

Cloning at runtime is helpful if you would like to use a (sub)task/module several times in your application.
Let's assume, one has a (sub)task/module which is responsible for creating a list of data object references based on specific input parameters. Cloning means in this context that one creates a second identical (sub)task but with different input parameters.

Such a mechanism is especially helpful for the realization of high level analysis tools. One example would be a task which creates a list of composite particle candidates. With the steering parameters you can define e.g. the name of the particle list, choose the fit algorithm + constraints and define the selection criteria for the individual (cloned) tasks.

Ciao,
Bertram.

Subject: Re: compilation of macros
Posted by Elwin Dijck on Fri, 04 Dec 2009 15:52:40 GMT
View Forum Message <> Reply to Message

Thanks!

There is now an example of how I compiled macros with g++ in the

macro/run/compile_example/ directory. There you'll find a Makefile and source code that compile something like sim_complete_tpc.C into a stand-alone executable that accepts parameters from the command-line. The final linking step should now work with the newest PandaRoot revision.

Best regards,
Elwin Dijck

Subject: Re: compilation of macros
Posted by Florian Uhlig on Mon, 07 Dec 2009 13:38:44 GMT
View Forum Message <> Reply to Message

Hi Bertram

Quote:Cloning at runtime is helpful if you would like to use a (sub)task/module several times in your application.
Let's assume, one has a (sub)task/module which is responsible for creating a list of data object references based on specific input parameters. Cloning means in this context that one creates a second identical (sub)task but with different input parameters.

I think you don't understand what rootcint and the framework can do. What you requested is already there. For example have a look at eventDisplay.C in the macro/run directory of pandaroot. There you will find an example which shows that you add many times one task with differnt parameters in the same macro. If I understand you correctly this is exactely what you're are looking for.

Ciao

Florian

Subject: Re: compilation of macros
Posted by Bertram Kopf on Mon, 07 Dec 2009 13:55:05 GMT
View Forum Message <> Reply to Message

Hi Florian,

Quote:
I think you don't understand what rootcint and the framework can do. What you requested is already there. For example have a look at eventDisplay.C in the macro/run directory of pandaroot. There you will find an example which shows that you add many times one task with differnt parameters in the same macro. If I understand you correctly this is exactely what you're are looking for.

Yes, of course! But how is it possible to communicate with or clone subtasks which are not directly constructed in the top macro?

Best regards,

Bertram.

Subject: Re: compilation of macros
Posted by Mohammad Al-Turany on Mon, 07 Dec 2009 14:19:38 GMT
View Forum Message <> Reply to Message

Hi Bertram,

just for my curiosity, do you have any real use case for this? I thought you do not want to use macros at all, you wanted simply to compile these macros, and now you want to control even the sub-tasks (Which is by the way possible!) from the macros. It would be really great if all this would have a technical background and use cases and not simply like it is now!

regards

Mohammad

Subject: Re: compilation of macros
Posted by Bertram Kopf on Mon, 07 Dec 2009 16:24:42 GMT
View Forum Message <> Reply to Message

Hi Mohammad,

Quote:
just for my curiosity, do you have any real use case for this? I thought you do not want to use macros at all, you wanted simply to compile these macros, and now you want to control even the sub-tasks (Which is by the way possible!) from the macros. It would be really great if all this would have a technical background and use cases and not simply like it is now!

the full reconstruction and analysis application for example will exist of a huge amount of individual tasks. To keep everything under control, it would be good/mandatory to set it up via an hierarchical structure, i.e. with sequences consisting of subtasks and subsubtasks. And such tasks are needed to be controled via an interpreter. Great, if this is possible with the existing application framework!
As I already mentioned before, one use case would be the high level analysis tools.

Cheers,
Bertram.

Subject: Re: compilation of macros
Posted by Johan Messchendorp on Mon, 07 Dec 2009 18:16:19 GMT
View Forum Message <> Reply to Message

Dear Bertram and others,

It is really great that Bertram thinks about the very important high-level analysis tools and its corresponding framework. From the physics book activities, Bertram has gained lots of experiences in this direction, in particular with a good overview of the needs of users. Furthermore, we indeed need these higher-level tools/framework and at the moment this is far from complete (mostly due to lack of manpower in this direction). I would therefore encourage any thought in this direction!

The question is what the best strategy is to proceed here. Personally, I would like to see that such a high-level framework is developed along the lines within the present framework. For two reasons: first of all, it would be good to keep things transparent for developers and users, and secondly, it is simply much more efficient to build on something we already have than to re-invent the wheel. We really do not have any resources to follow more lines (and we will also not have this in the future).

Following this discussion in the forum, I understand that - in principle - the development of a higher-level analysis framework with sub-sub-sub tasks etc is possible within the existing framework. My request is therefore to first start looking into what is there and how this can be used to continue the higher-level analysis developments. For me this is the most logical approach. But if someone has another point-of-view on this, let me know....

Kind wishes,

Johan.

Subject: Re: compilation of macros
Posted by Elwin Dijck on Thu, 07 Jan 2010 14:49:20 GMT
View Forum Message <> Reply to Message

At the moment the compilation of macros doesn't work anymore because there is again a declared but unimplemented public function. In change 7385, the implementation of PndMvdDigiStrip::PndMvdDigiStrip(Int_t, Int_t, TString, Int_t, Int_t, Double_t, Int_t) was removed, but it's still declared and actually also used (in PndMvdConvertApv). Could someone repair this?

Subject: Re: compilation of macros
Posted by Tobias Stockmanns on Thu, 07 Jan 2010 15:18:41 GMT
View Forum Message <> Reply to Message

Bug is fixed. Please check out the latest version of PndMvdDigiStrip.cxx

---