# Include-what-you-use with the LLVM

Rolf Seuster (MPI)
Future Computing in Particle Physics
Edinburgh, 15th June 2011

- Introduction
- Include-what-you-use
- Webdisplay, Performance, CPU consumption and quality of reports
- Conclusion

# Disclaimer

- Include-what-you-use is not about speeding up code, but about code quality
  - possible to clean up dependencies between packages and speed up compilation, results should be unchanged.
- For fun, tried to compile few packages with clang, but failed
  - interference system compiler on slc5 with compiler used by ATLAS (4.1.2 vs 4.3.2)
    - despite all patches mention in mailing list(s), clang picked up header files from system compiler

# Introduction IWYU

- Include-What-You-Use is a compiler extension for clang compiler based on LLVM (low level virtual machine).
- scans the code to detect what header files are included and which ones should have been / should not have been included
- if possible, IWYU recommends to replace include with a forward-declaration.
- obviously, complicated task, and software is “early beta quality” according to authors – might still be useful to use within ATLAS
- links:
  http://code.google.com/p/include-what-you-use/
  http://clang.llvm.org/index.html
  http://llvm.org/

# Introduction IWYU in ATLAS

- few small modifications needed to be done for the scanning of the ATLAS repository, which might result in slightly obscured reports:
  - quotes changes to brackets to avoid many add/remove suggestions of the same include files
  - for scanning header files, location of the header files of the compiled package changed to "src". In this pass, IWYU crashed, so a two-pass compilation is needed, headers and source files

# Introduction IWYU

- recently scanned all atlas projects. The scan was done in about 40 hours on a 16 core machine with 16 processes in parallel. The packages were checked out and processed locally.
- results now being displayed on dedicated webpage (html + xml)

# Introduction - Installation

- Installation was not 100% straight forward, as instructions from LLVM/clang and IWYU are slightly contradicting themself w.r.t. build directory. LLVM recommends to have separate build and source directories whereas IWYU seems to require it to compile.
- Solution: unpack LLVL, then unpack IWYU into LLVM source, then compile in separate build dir.

# IWYU, comments

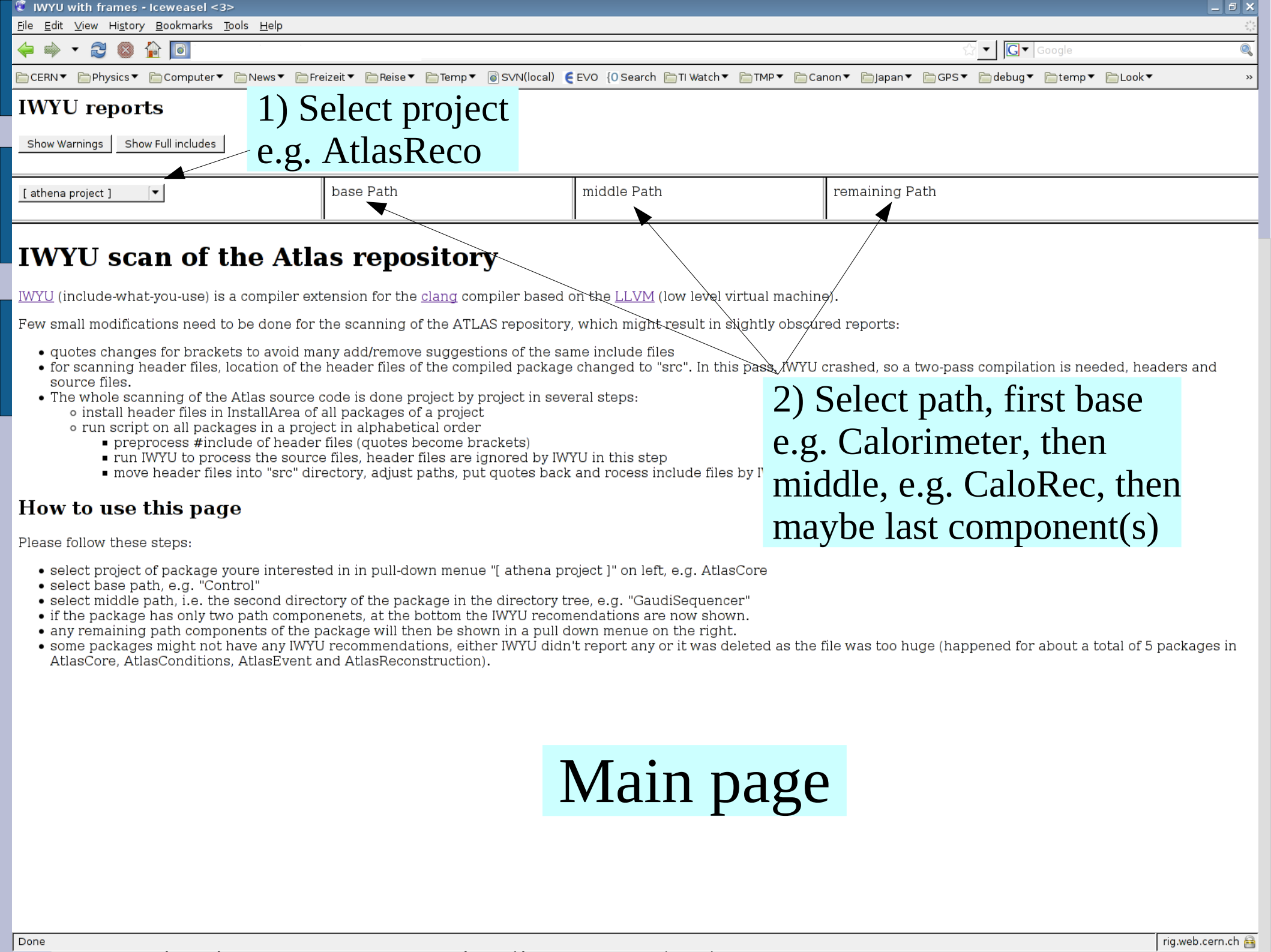
- IWYU apparently doesn't see icc files
  (might be related to crash when scanning headers ?!?)
  - might gets lots of false recommendations
- should ignore autogenerated files, lots of false recommendations from there
- some files are huge, biggest one over 3GB
  often: unused template parameters inside boost etc.
- otherwise, I haven't seen serious problems with IWYU, no reason not to take recommendations serious

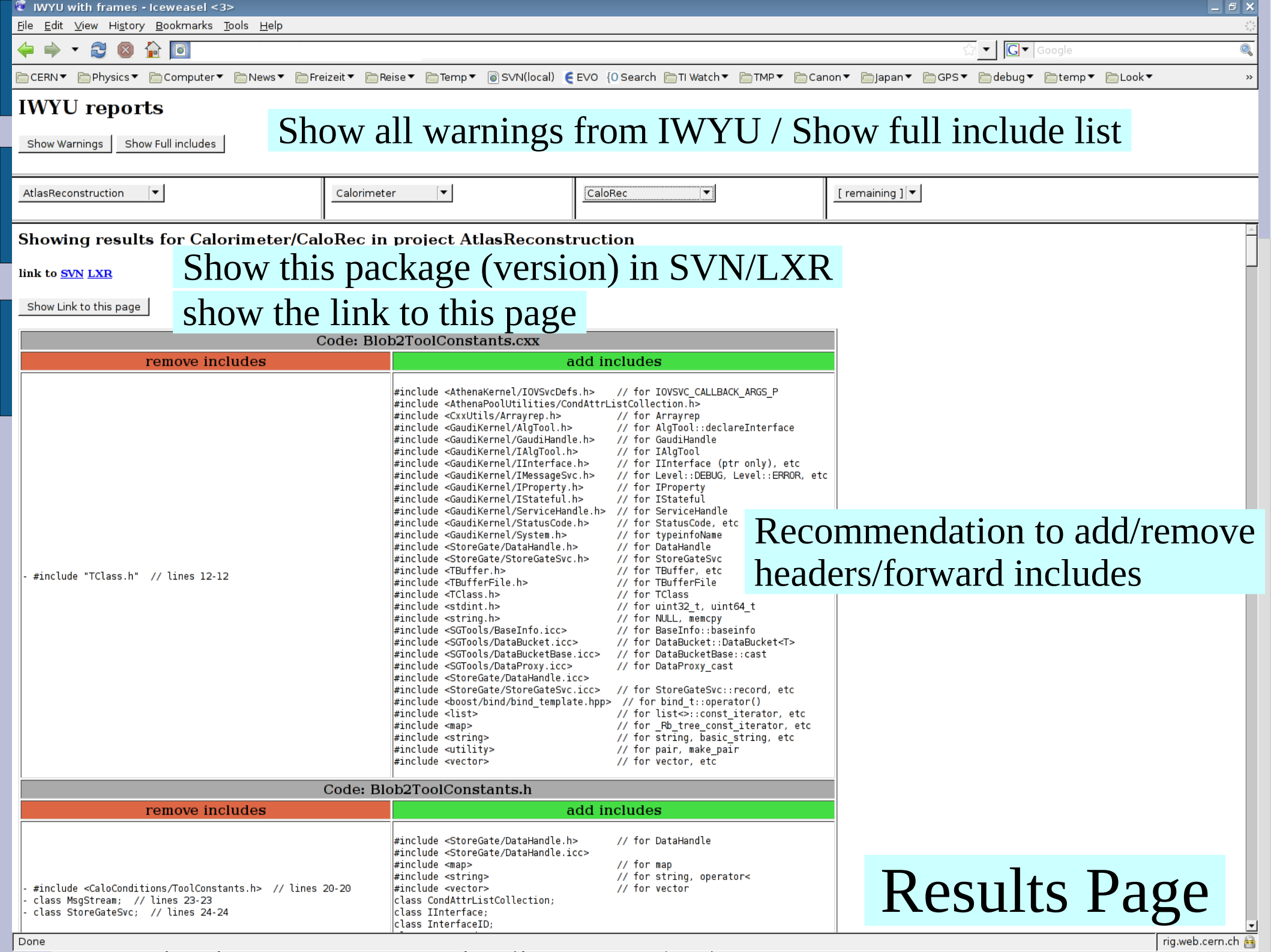
# CPU consumption of the scan

- ATLAS code consists of several 'projects', scan was done for these projects:

| Project | number of packages | time to completescan |
|---|---|---|
| DetCommon | 17 | 00:07:38 |
| AtlasCore | 191 | 02:20:32 |
| AtlasConditions | 223 | 03:15:23 |
| AtlasEvent | 354 | 06:51:48 |
| AtlasReconstruction | 471 | 08:27:39 |
| AtlasTrigger | 222 | 05:31:30 |
| AtlasAnalysis | 349 | 11:20:17 |
| AtlasSimulation | 214 | 03:11:21 |

# Display on the webpage


IWYU with frames - Iceweasel <3>
File Edit View History Bookmarks Tools Help
IWYU reports
Show Warnings
Show Full includes
1) Select project e.g. AtlasReco
[ athena project ]
base Path
middle Path
remaining Path
IWYU scan of the Atlas repository
IWYU (include-what-you-use) is a compiler extension for the clang compiler based on the LLVM (low level virtual machine).
Few small modifications need to be done for the scanning of the ATLAS repository, which might result in slightly obscured reports:
quotes changes for brackets to avoid many add/remove suggestions of the same include files
for scanning header files, location of the header files of the compiled package changed to "src". In this pass IWYU crashed, so a two-pass compilation is needed, headers and source files.
The whole scanning of the Atlas source code is done project by project in several steps:
install header files in InstallArea of all packages of a project
run script on all packages in a project in alphabetical order
preprocess #include of header files (quotes become brackets)
run IWYU to process the source files, header files are ignored by IWYU in this step
move header files into "src" directory, adjust paths, put quotes back and rocess include files by I
2) Select path, first base e.g. Calorimeter, then middle, e.g. CaloRec, then maybe last component(s)
How to use this page
Please follow these steps:
select project of package youre interested in in pull-down menue "[ athena project ]" on left, e.g. AtlasCore
select base path, e.g. "Control"
select middle path, i.e. the second directory of the package in the directory tree, e.g. "GaudiSequencer"
if the package has only two path componenets, at the bottom the IWYU recomendations are now shown.
any remaining path components of the package will then be shown in a pull down menue on the right.
some packages might not have any IWYU recommendations, either IWYU didn't report any or it was deleted as the file was too huge (happened for about a total of 5 packages in AtlasCore, AtlasConditions, AtlasEvent and AtlasReconstruction).
Main page
Done
rig.web.cern.ch


IWYU with frames - Iceweasel <3>
File Edit View History Bookmarks Tools Help
IWYU reports
Show Warnings
Show Full includes
Show all warnings from IWYU / Show full include list
AtlasReconstruction
Calorimeter
CaloRec
[ remaining ]
Showing results for Calorimeter/CaloRec in project AtlasReconstruction
link to SVN LXR
Show Link to this page
Show this package (version) in SVN/LXR
show the link to this page
Code: Blob2ToolConstants.cxx
remove includes
add includes
- #include "TClass.h" // lines 12-12
#include <AthenaKernel/IOVSvcDefs.h> // for IOVSVC_CALLBACK_ARGS_P
#include <AthenaPoolUtilities/CondAttrListCollection.h>
#include <CxxUtils/Arrayrep.h> // for Arrayrep
#include <GaudiKernel/AlgTool.h> // for AlgTool::declareInterface
#include <GaudiKernel/GaudiHandle.h> // for GaudiHandle
#include <GaudiKernel/IAlgTool.h> // for IAlgTool
#include <GaudiKernel/IInterface.h> // for IInterface (ptr only), etc
#include <GaudiKernel/IMessageSvc.h> // for Level::DEBUG, Level::ERROR, etc
#include <GaudiKernel/IProperty.h> // for IProperty
#include <GaudiKernel/IStateful.h> // for IStateful
#include <GaudiKernel/ServiceHandle.h> // for ServiceHandle
#include <GaudiKernel/StatusCode.h> // for StatusCode, etc
#include <GaudiKernel/System.h> // for typeinfoName
#include <StoreGate/DataHandle.h> // for DataHandle
#include <StoreGate/StoreGateSvc.h> // for StoreGateSvc
#include <TBuffer.h> // for TBuffer, etc
#include <TBufferFile.h> // for TBufferFile
#include <TClass.h> // for TClass
#include <stdint.h> // for uint32_t, uint64_t
#include <string.h> // for NULL, memcpy
#include <SGTools/BaseInfo.icc> // for BaseInfo::baseinfo
#include <SGTools/DataBucket.icc> // for DataBucket::DataBucket<T>
#include <SGTools/DataBucketBase.icc> // for DataBucketBase::cast
#include <SGTools/DataProxy.icc> // for DataProxy_cast
#include <StoreGate/DataHandle.icc>
#include <StoreGate/StoreGateSvc.icc> // for StoreGateSvc::record, etc
#include <boost/bind/bind_template.hpp> // for bind_t::operator()
#include <list> // for list<>::const_iterator, etc
#include <map> // for _Rb_tree_const_iterator, etc
#include <string> // for string, basic_string, etc
#include <utility> // for pair, make_pair
#include <vector> // for vector, etc
Recommendation to add/remove headers/forward includes
Code: Blob2ToolConstants.h
remove includes
add includes
- #include <CaloConditions/ToolConstants.h> // lines 20-20
- class MsgStream; // lines 23-23
- class StoreGateSvc; // lines 24-24
#include <StoreGate/DataHandle.h> // for DataHandle
#include <StoreGate/DataHandle.icc>
#include <map> // for map
#include <string> // for string, operator<
#include <vector> // for vector
class CondAttrListCollection;
class IInterface;
class InterfaceID;
Results Page
Done
rig.web.cern.ch

# Some Results from the scan

## 1st example: JetRec

- package JetRec
- mostly OK, but
  - gcc accepts forward declaration of class in template ToolHandle<class>
  - false recommendation to forward declare TTree (using TTree::Branch()) in header file:

```
template<typename T>  void addBranch(TTree* pTree, const std::string& bName, T*& obj)
{
  obj = new T();
  pTree->Branch(bName.c_str(),&obj);
}
```

**Code: CBNTAA_JetToolBase.cxx**

| remove includes | add includes |
|---|---|
| - #include <JetEvent/JetCollection.h>  // lines 5-5<br>- #include "TTree.h"  // lines 3-3 | #include <GaudiKernel/IAlgTool.h>  // for IAlgTool<br>#include <GaudiKernel/IMessageSvc.h>  // for Level::WARNING, etc<br>#include <GaudiKernel/IProperty.h>  // for IProperty<br>#include <GaudiKernel/IStateful.h>  // for IStateful<br>#include <GaudiKernel/MsgStream.h>  // for MsgStream, operator<<, etc<br>#include <GaudiKernel/StatusCode.h>  // for StatusCode, etc<br>#include <string>  // for string<br>class IInterface;<br>class JetCollection;<br>class TTree; |

**Code: CBNTAA_JetToolBase.h**

| remove includes | add includes |
|---|---|
| | #include <GaudiKernel/StatusCode.h>  // for StatusCode, etc<br>class IInterface;<br>class TTree; |

# Some Results from the scan
## 2nd example: TrackParticleSlimming

- package TrackParticleSlimming
  - mostly OK, replace several includes with forward declarations
    many includes only needed in cxx file
  - even points to bug in c++ code: mix MsgStream w. iostream

**Code: TrackSlimmingHdlr.cxx**

| remove includes | add includes |
|---|---|
| | #include <AthenaKernel/MsgStreamMember.h> // for operator<<, etc<br>#include <AthenaKernel/getMessageSvc.h> // for CreateOptions::Eager<br>#include <GaudiKernel/IMessageSvc.h> // for Level::DEBUG, Level::ERROR<br>#include <GaudiKernel/MsgStream.h> // for operator<<, MsgStream, etc<br>#include <Particle/TrackParticle.h> // for TrackParticle<br>#include <TrkParameters/MeasuredTrackParameters.h><br>#include <TrkParameters/Perigee.h> // for Perigee<br>#include <TrkParametersBase/ParametersBase.h> // for ParametersBase<br>#include <stddef.h> // for NULL<br>#include <algorithm> // for swap<br>#include <ostream> // for endl<br>#include <vector> // for vector, vector<>::iterator<br>class INamedInterface;<br>namespace std { class type_info; } |

**Code: TrackSlimmingHdlr.h**

| remove includes | add includes |
|---|---|
| - #include <Particle/TrackParticle.h> // lines 12-12<br>- #include <Particle/TrackParticleContainer.h> // lines 11-11<br>- #include <TrkEventPrimitives/FitQuality.h> // lines 16-16<br>- #include <TrkEventPrimitives/ParamDefs.h> // lines 19-19<br>- #include <TrkParameters/Perigee.h> // lines 18-18<br>- #include <TrkParametersBase/ParametersBase.h> // lines 13-13<br>- #include <TrkTrack/Track.h> // lines 17-17<br>- #include <TrkTrack/TrackCollection.h> // lines 15-15<br>- #include <TrkTrackSummary/TrackSummary.h> // lines 14-14<br>- #include <iostream> // lines 8-8<br>- #include <typeinfo> // lines 6-6<br>- #include <vector> // lines 5-5 | #include <string> // for string<br>class INamedInterface;<br>namespace Rec { class TrackParticle; }<br>namespace Trk { class ParametersBase; }<br>namespace std { class type_info; } |

# Discussion

- to be really useful, needs to be combined with other approaches, e.g.
  - reduce “dead” code from reconstruction perspective:
    - remove code not run during reconstruction and move code of into other packages
    - reduce code size
    - better locality of code, though cachegrind reports only 1.7% / 1.2% cache misses for intructions / data
- google aims at combining this with
  - precompiled / preparsed headers

# Conclusion

- aim is to clean up includes and then clean up cmt requirements files (private/public)
- reduce package dependencies and speed up compilation (wishful thinking ?)

- Should have close look what 'professionals' are doing
  - google can safe real money getting their code in shape – and we can learn from them (tcmalloc)