Table of Contents

Running the algorithm Athena Hello World....................1

Introduction....................2

Copying the HelloAlg algorithm to HelloAlgNEW and making the necessary changes....................3

Compiling the package hosting the two algorithms....................5

Preparing to run the two algorithms at the same time....................6

Running the two algorithms at the same time....................7

Running the algorithm Athena Hello World

Introduction

This page describes how to run the algorithm Hello World. This is already described in the general ATLAS Twiki Running Athena HelloWorld tutorial and should work out of the box from there. This is my step by step implementation on a lxplus machine at CERN. That way anyone can reproduce it easily in a controlled environment.

Work in progress ...

If you have not yet done so, the first step is to checkout the Hello World package described in the official ATLAS Twiki

This page describes how to add a new algorithm to an existing package and how to run the two algorithms at the same time. The easiest way is to copy an existing algorithm and make modifications. We will copy the HelloWorld algorithm to a new one, we will modify the new one slightly and we will run both at the same time.

Please also see a discussion at https://twiki.cern.ch/twiki/bin/view/AtlasProtected/PhysicsAnalysisWorkBookAlgorithmRel15

First, log into your account and set up CMT (as described in WorkBookSetAccount). Assuming you are running in a folder with the same name as the Athena version and located in the ~/testarea, if you have a bash shell define this variable which will help as move from a folder to another folder easily while allowing us to use the same copy paste commands for different versions of Athena.

```
export ATHENA_VERSION=17.0.5.5.2
cd ~/testarea/$ATHENA_VERSION
asetup $ATHENA_VERSION,here
```

If you have not yet done so, the first step is to checkout the Hello World package described in the official ATLAS Twiki Running Athena HelloWorld

Copying the HelloAlg algorithm to HelloAlgNEW and making the necessary changes

```
cd ~/testarea/$ATHENA_VERSION/Control/AthenaExamples/AthExHelloWorld/src
cp HelloAlg.cxx HelloAlgNEW.cxx
```

Then, replace all appearances of `HelloAlg` with within `HelloAlgNEW` within `HelloAlgNEW.cxx` or equivalently:

```
sed -i 's/HelloAlg/HelloAlgNEW/g' HelloAlgNEW.cxx
```

Now let's add the string "NEW" in the message statements. First, in the initialize() method replace

```
  ATH_MSG_INFO ("initialize()")
```

with

```
  ATH_MSG_INFO ("initialize() NEW")
```

then in the execute() method replace

```
  ATH_MSG_INFO ("execute()")
```

with

```
  ATH_MSG_INFO ("execute() NEW")
```

then in the finalize() method replace

```
  ATH_MSG_INFO ("finalize()")
```

with

```
  ATH_MSG_INFO ("finalize() NEW")
```

Now something particular to the HelloAlg algorithm. In the beginning of the .cxx file there is an << operator defined outside the class and the compilation would fail if we keep this code in both files. So let's comment these lines in HelloAlgNEW.cxx, where you should have this:

```
/////////////////////////////////////////////////////////////////////////////
// FIXME Looks like we need operator<<( ostream&, pair<double, double > ) for gcc41
//std::ostream& operator<<( std::ostream& s, std::pair<double, double > p )
//{
//s << p.first << " " << p.second;
//return s;
//}
//we should comment it, as it is already in the HelloAlg.cxx and it gives compilation error
```

Then, replace all appearances of `HelloAlg` with within `HelloAlgNEW` within `HelloAlgNEW.h` or equivalently:

```
cp HelloAlg.h HelloAlgNEW.h
sed -i 's/HelloAlg/HelloAlgNEW/g' HelloAlgNEW.h
```

We also need to modify the preprocessor lines to have them unique for our new package: for example, replace:

```
#ifndef ATHEXHELLOWORLD_HELLOALG_H
```

```
#define ATHEXHELLOWORLD_HELLOALG_H 1
```

with:

```
#ifndef ATHEXHELLOWORLD_HELLOALG_NEW_H
#define ATHEXHELLOWORLD_HELLOALG_NEW_H 1
```

Then we need to add a line for the new algorithm for every line for the old algorithm in the components directory.

```
cd components
emacs -nw AthExHelloWorld_entries.cxx
```

The file will look like this

```
#include "../HelloAlg.h"
#include "../HelloAlgNEW.h"
#include "../HelloTool.h"

#include "GaudiKernel/DeclareFactoryEntries.h"

DECLARE_ALGORITHM_FACTORY( HelloAlg )
DECLARE_ALGORITHM_FACTORY( HelloAlgNEW )
DECLARE_TOOL_FACTORY( HelloTool )

DECLARE_FACTORY_ENTRIES(AthExHelloWorld) {
DECLARE_ALGORITHM( HelloAlg )
DECLARE_ALGORITHM( HelloAlgNEW )
DECLARE_TOOL( HelloTool )
}
```

Compiling the package hosting the two algorithms

Now let's compile the AthExHelloWorld package, which contains now two algorithms

```
cd ../../cmt
cmt config
gmake
```

Preparing to run the two algorithms at the same time

Now let's modify the job options so that we run both algorithms at the same time. Let's copy the option in a new file and modify only this copy, so that we can always compare with the original one for reference.

```
cd ~/testarea/$ATHENA_VERSION/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
cp HelloWorldOptions.py HelloWorldOptionsNEW.py
emacs -nw HelloWorldOptionsNEW.py
```

Replace these lines

```
# Add top algorithms to be run
from AthExHelloWorld.AthExHelloWorldConf import HelloAlg
job += HelloAlg( "HelloWorld" )   # 1 alg, named "HelloWorld
```

with these lines

```
# Add top algorithms to be run
from AthExHelloWorld.AthExHelloWorldConf import HelloAlg, HelloAlgNEW
job += HelloAlg( "HelloWorld" )   # 1 alg, named "HelloWorld
job += HelloAlgNEW( "HelloWorldNEW" )   # 1 alg, named "HelloWorldNEW"
```

Running the two algorithms at the same time

We can now run again, without a need to recompile the UserAnalysis package

```
athena.py HelloWorldOptionsNEW.py
```

We can check that we ran both algorithms together if we have an output line containing "TEST" and just under it another line containing "TEST_NEW".

Major updates:
-- AdrianBuzatu - 25-Jan-2012

-- AdrianBuzatu - 27-Feb-2012

This topic: Sandbox > AdrianBuzatuAthenaHelloWorldRunning
Topic revision: r1 - 2012-02-27 - unknown