Introduction to CMSSW:L3

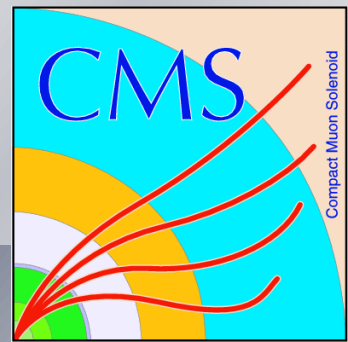
Presented by

DR. MOHAMMED ATTIA MAHMOUD

-PhD, Fayoum University, Egypt and Antwerp University, Belgium.

-Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.

-FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.



Outlines

- How to write your own analyzer
- Configuration file

How to write your own analyzer

- You have to have CERN account, for login to lxplus

  ssh –Y OR X username@lxplus.cern.ch

- Listing the available CMSSW:

  scram list

- For changing to new architecture

  setenv ARCH <your-new-arch>
  Example: setenv SCRAM_ARCH slc6_amd64_gcc481

- Choosing CMSSW By using this command:

  cmsrel CMSSW_X_Y_Z

- Apply cms environment :

  cmsenv

How to write your own analyzer

- **First, create a subsystem area. The actual name used for the directory is not important, we'll use First_analysis.**

```
mkdir First_analysis
cd First_analysis
```

- **Create the "skeleton" of an EDAnalyzer module**

```
mkedanlzr DemoAnalyzer
```

- **Compile the code:**

```
cd DemoAnalyzer
scram b
```

Configuration file

Always need to import this

```
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")

process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(-1) )

process.source = cms.Source("PoolSource",
                            # replace 'myfile.root' with the source file you want to use
                            fileNames = cms.untracked.vstring(
        'file:/afs/cern.ch/cms/Tutorials/TWIKI_DATA/TTJets_8TeV_53X.root'
        )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer'
                              )

process.p = cms.Path(process.demo)
```

A config must always create a cms.Process called process. The label (MAIN in this case) will be applied to any objects added to the root files

Create a data source, a list of files in this case.

Number of events to process. -1 means process all events in the input files. NB. When submitted jobs to the grid this parameter will be overridden automatically

The last step is usually to define one or more paths containing the sequence of modules to actually run

Define some modules. The first argument should be the name of the module as defined in the C++ class (usually the class name). The subsequent named arguments define the parameter set for that module.

BuildFile

SCRAM uses a file called CMS.BuildFile in each package directory which describes what the package will produce and what dependencies the package has. Consider the following CMS.BuildFile from the tutorial:

```
<use name="FWCore/Framework"/>
 <use name="FWCore/PluginManager"/>
<use name="FWCore/ParameterSet"/>
<use name="DataFormats/TrackReco"/>
 <use name="CommonTools/UtilAlgos"/>
<flags EDM_PLUGIN="1"/>
```

The first part of the CMS.BuildFile tells SCRAM what packages or external libraries (e.g., **FWCore/Framework**) are needed to build this package. The <flags> line is needed because this package contains a framework module (in this case, your analyzer) which must be registered with the plugin system .

File.cc (DemoAnalyzer)

This file is located in plugin directory in the same place of BuildFile, please go to this directory by using cd command. Open it with any editor like pico, vi, vim, gedit, medit, ..

```
#include "DataFormats/TrackReco/interface/Track.h"
#include "DataFormats/TrackReco/interface/TrackFwd.h"
#include "FWCore/MessageLogger/interface/MessageLogger.h"
```

- Edit the method analyze which starts with

```
DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
```

and put the following lines below using namespace edm;

```
Handle<reco::TrackCollection> tracks;
iEvent.getByLabel("generalTracks", tracks);
LogInfo("Demo") << "number of tracks "<<tracks->size();
```

```
for (reco::TrackCollection::const_iterator it = tracks->begin(); it !=
tracks->end(); it++){
const reco::Track &track = **it;

OR

const reco::Track* track = &(*it);

YOU CAN PUT ANY CUT here by using if statement
 if(…….. &&  ……..|| ){
        Track_pt   -> Fill(track->pt());
        track_phi  ->Fill(track->phi());

                }
      }
```


Thanks!