```
#ifndef INC_D0CHUNKANALYZE
#define INC_D0CHUNKANALYZE
////////////////////////////////////////////////////////////////////////////////
//
//
// File: D0ChunkAnalyze.hpp
//
// Purpose:  perform simple analysis at the D0 physics objects level
//
// Created:  07/24/2002 Marco Verzocchi
//
// History:
//
//
////////////////////////////////////////////////////////////////////////////////

// Dependencies (#includes)

#include "edm/Event.hpp"
#include "edm/TKey.hpp"
#include "edm/Tag.hpp"

#include "framework/Package.hpp"
#include "framework/hooks/Process.hpp"
#include "framework/hooks/Analyze.hpp"
#include "framework/hooks/JobSummary.hpp"

#include "ErrorLogger/ErrorLog.h"

#include "HepTuple/HepRootFileManager.h"

#include "em_evt/EMparticleChunk.hpp"
#include "jet_evt/JetChunkSelector.hpp"

#include "analysis_utilities/D0TriggerSelector.hpp"

#include <vector>

#include "TH1.h"
#include "TH2.h"
#include "TTree.h"

////////////////////////////////////////////////////////////////////////////////

namespace D0example {

```

```
class D0ChunkAnalyze : public fwk::Package,  public fwk::Process,
                       public fwk::Analyze,  public fwk::JobSummary,
                       public fwk::FileOpen, public fwk::FileClose {

  public :

    // Constructor/Destructor
    D0ChunkAnalyze(fwk::Context* context);
    ~D0ChunkAnalyze();

    // Overridden hook methods
    fwk::Result processEvent(edm::Event &event);
    fwk::Result analyzeEvent(const edm::Event &event);
    fwk::Result fileOpen(const fwk::FileInfo &fileinfo);
    fwk::Result fileClose(const fwk::FileInfo &fileinfo);
    fwk::Result jobSummary();

    // Overridden package methods
    std::string packageName() const {return package_name();}
    static const std::string package_name() {return "D0ChunkAnalyze";}
    static const std::string version() {return "$Id: D0ChunkAnalyze.hpp,v 1.2 2002/08/16 16:16:45 mverzocc Exp $";}

  private:

    // Method to write the histograms in a ROOT file.
    void SaveHistograms();

    // Print debug information:
    // _debug              turn on or off the debug information.
    bool _debug;

    // Do stupid things for educational purposes (D0 software tutorial):
    // _tutorial           turn on all the stupid things.
    bool _tutorial;

    // Buffers for RCP parameters:
    // _emKey              key for accessing the required EM particle chunk;
    // _jetCSel            jet chunk selector;
    edm::TKey<emid::EMparticleChunk> _emKey;
    jetid::JetChunkSelector _jetCSel;

    // Tool for performing trigger selection.
    D0analysis::D0TriggerSelector *_selectTriggers;

    // Histograms:
    // _do_histograms      controls whether histograms are filled or not;
```

```
    // _histo_saved        controls whether histograms have been written out or not;
    // _hepMgr             pointer to the ROOT method for handling I/O.
    bool _do_histograms;
    bool _histo_saved;
    HepRootFileManager *_hepMgr;

    // List of all the histograms used in the code (ROOT histograms are
    // used directly instead of using the HepTuple interface).

    // Electron histograms
    TH1F *_h1numEle;
    TH2F *_h2emFvsIso;
    TH1F *_h1EoverP;
    TH1F *_h1numTkMatch;

    // Muon histograms
    TH1F *_h1numMuo;
    TH1F *_h1muoPt;
    TH2F *_h2muGlobLocal;

    // Jet histograms
    TH1F *_h1numJet;
    TH1F *_h1jetPt;

    // Missing ET histograms
    TH1F *_h1SET;
    TH1F *_h1MET;

    // Tracks histograms
    TH1F *_h1trkpT;
    TH1F *_h1trkPhi;
    TH1F *_h1trkDip;
    TH1F *_h1trkSigd0;
    TH1F *_h1trkSMThits;
    TH1F *_h1trkCFThits;
    TH1F *_h1trkZ0;
    TH1F *_h1numVtxMatch;
    TH2F *_h2trkVtxXY;

    // Simple ROOT tuple.
    TTree *_ttreeTracks;
    Int_t _nTracks;
    Float_t _fpTTracks[100];
    Float_t _fetaTracks[100];
    Float_t _fphiTracks[100];

```

```
      // Event counters:
      // _processed_events   total number of events processed;
      // _triggered_events   total number of events passing the trigger requirements;
      // _selected_events    total number of events passing the user selection.
      int _processed_events;
      int _triggered_events;
      int _selected_events;

      // Event tags:
      // _event_tag          tag used for selecting events.
      edm::Tag _event_tag;

      // Error log
      ErrorLog errlog;

  }; // class D0ChunkAnalyze

} // namespace D0example
#endif // INC_D0CHUNKANALYZE
```