HEP Analysis, C++ & ROOT

Stewart Boogert

Ph.D : ZEUS

Post Doc : Linear Collider

Aim of introduction : Operational introduction to using ROOT for HEP analysis. No experiment/project specific code/examples

Wider aim : To get you working and productive as soon as possible opposed to spending months reinventing the wheel

Why does this course exist?

- Computing for HEP is significantly different from home/office computing
  - Linux based, more control, flexibility and utility
  - CPU/Memory/Disk space intensive
  - In house software tools
- Typically
  - Design software from near scratch for analysis or research
    - Many applications/experiments require the same tools
  - Rely on HEP packages, libraries to provide functionality
    - ROOT
    - HBOOK
    - MatLab, Mathematica, LabView
- Personally
  - I get asked loads of questions from 1st years
    - Same questions, just applied to different contexts
    - Can’t be bothered to answer any more, hence this introduction

HEP Analysis

- Fundamentals of high energy physics analysis
  - Pre-select events (from data storage)
  - Analysis (Write program to look at each event individually to calculate event properties)
  - Cosmetics (take histograms, graphs, numbers from Analysis and present them, making plots, tables etc)
- Pre-selection
  - Experiment specific
  - Output usually in form of .root .hbook files and contain distilled information of the selected events
- Analysis
  - Three stages
    - Initialisation (set parameters, create histograms, define files to be loaded, etc)
    - Event looping (look at each event in turn and perform calculations and fill histograms)
    - Termination (perform fits or other calculations, write histograms, graphs to file)

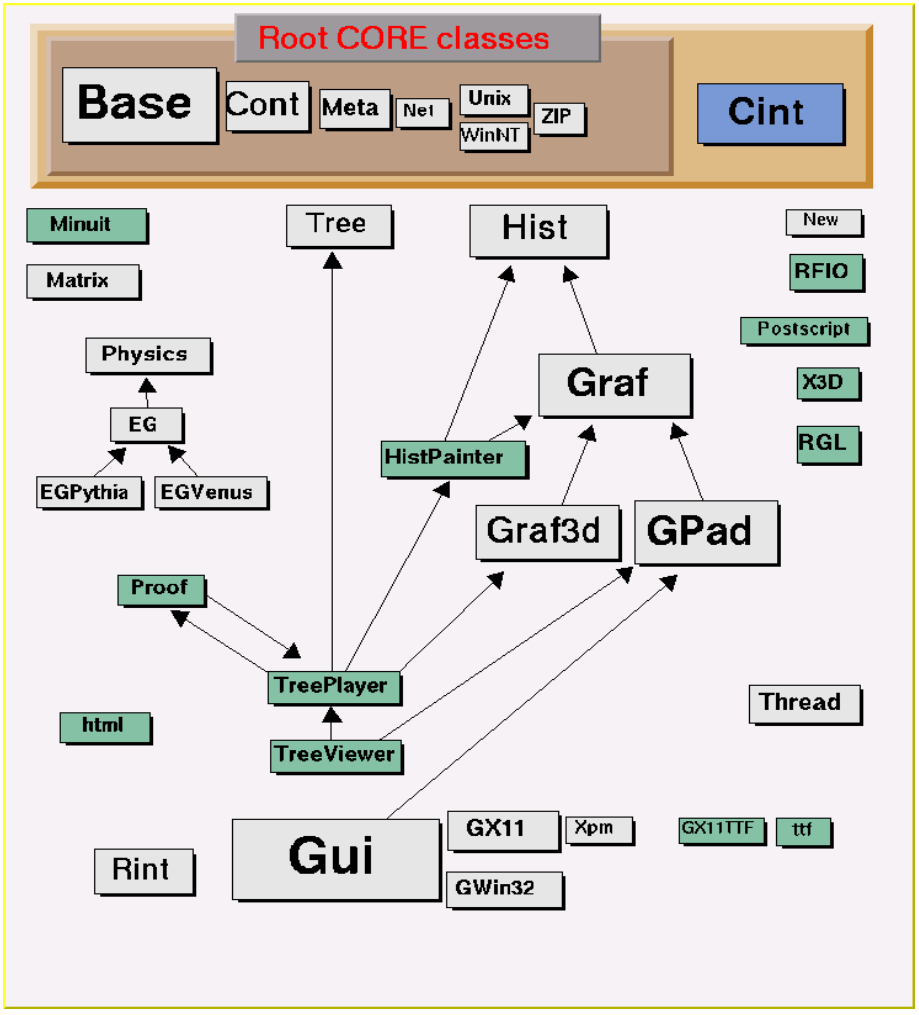
HEP data structures

- Three possible scenarios discussed here:
  - ASCII (text) file with columns of numbers
  - Hbook file, containing ntuple
  - Root file, containing trees
- Text
  - Most inflexible, useful for quick and dirty results, very difficult for large amounts of experimental data.
  - Used frequently in the lab
- Hbook
  - Legacy file format, from CERN, but still widely used in HEP
  - ROOT provides similar functionality
- ROOT
  - Current analysis framework
  - Object orientated (C++)

What is ROOT (1)?

- ROOT is basically a class library (www.dictionary.com)
  - Library :
    - 1) A place in which literary and artistic materials, such as books, periodicals, newspapers, pamphlets, prints, records, and tapes, are kept for reading, reference, or lending.
    - 5) A collection of recorded data or tapes arranged for ease of use.
  - Class :
    - 1) A set, collection, group, or configuration containing members regarded as having certain attributes or traits in common; a kind or category.
  - Classes are a programming construction which, among other things, allows you to use large amounts of other peoples code.

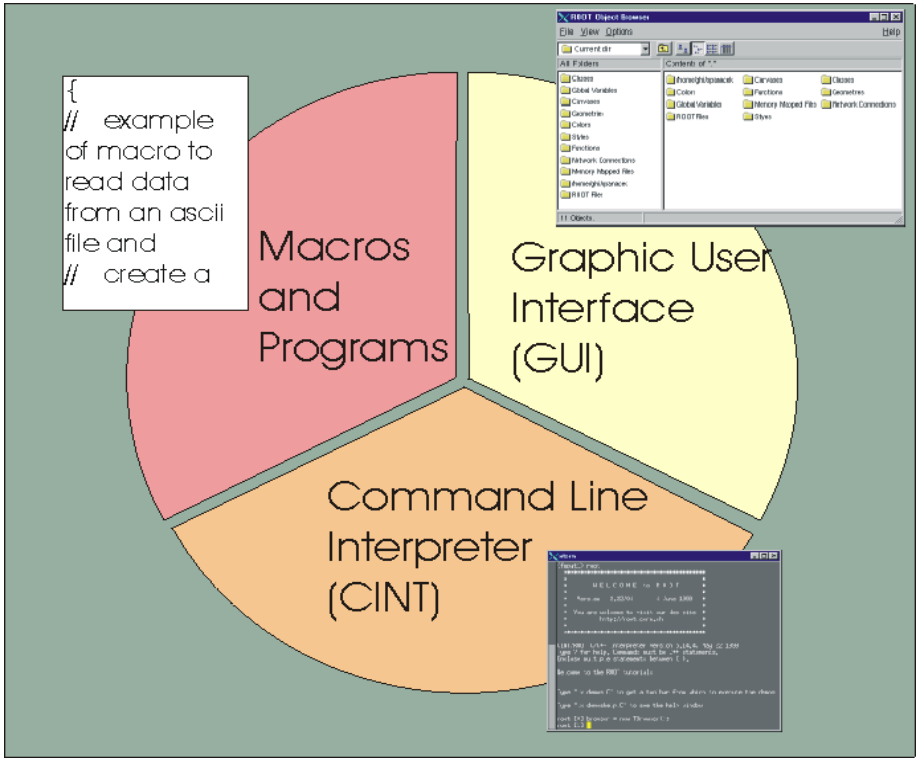
What is ROOT (2)?



- Histograms
  - 1D, 2D & 3D
- Graphs
  - Errors, multiple graphs
- Graphics
  - 2D/3D
- Collections
  - Arrays
- User interface
  - Graphical :Buttons, menus etc
  - Command line : CINT
- Data storage IO
  - Write complicated data structures to disk
  - Sockets/networking

Interface/Interaction with ROOT

- Three main methods to interact with ROOT
  - Write C++ and link against ROOT libraries
    - Good for large programs/analysis
    - Terrible for small jobs
  - GUI
    - Quick to display histograms and modify aesthetics
    - No good for complicated problems
  - Macros
    - CINT, C++ interpreter
    - Type C++ into ROOT and root evaluates it.
    - Best of all worlds

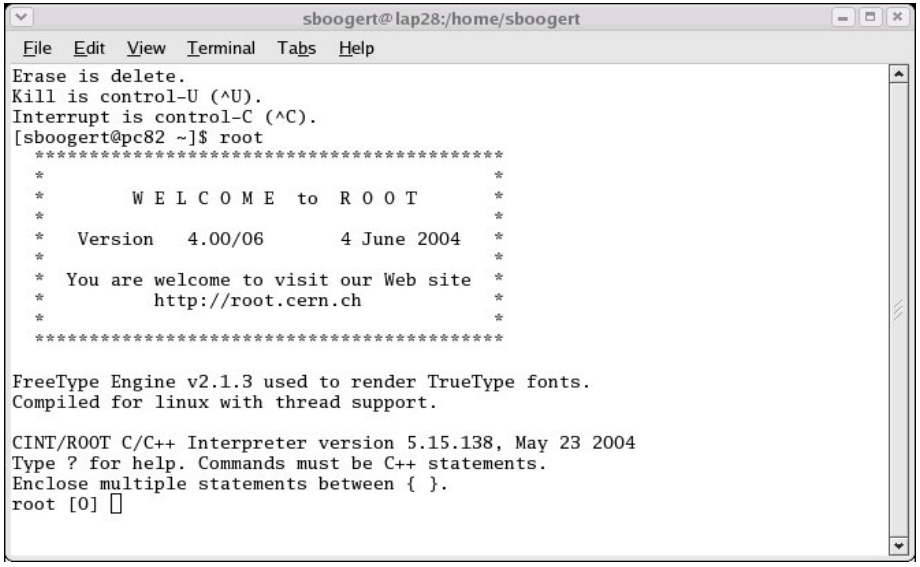


Configure ROOT

- ROOT is installed on most HEP computer clusters
  - To configure
  - Set variables :
    - ROOTSYS to the directory of ROOT installation you wish to use
    - PATH to include the directory where the executable components are kept
    - LD_LIBRARY_PATH to include the directory where the library files are kept (see Mark Lancasters talk about compilation and libraries)
  - Example in /unix/lc/software/lcsetup.sh (or lcsetup.csh)
- Should be able to start ROOT by typing root into the command line

Command line interface

- Start root
  - Get command line interface (CINT)
  - Can type any valid C++
  - {} are used for multiple lines of code
- Apart from C++ code, ROOT command begin with . So
  - .L load macro
  - .x execute macro
  - .h help of CINT
- Can start interactive components
  - new TBrowser();
  - new TCanvas();



Little bit of C++

- Basic types
  - Integers, float, double, char etc
  - Operations defined + - * sin(double)
  - What about more complicated types
- Classes/Objects
  - Like basic types
  - Construction must be defined
  - Operations/methods must be defined
  - Construction
    - `TH1D *h = new TH1D("h","h",100,0,100);`
    - `TH1D h = TH1D("h","h",100,0,100);`
  - Methods
    - `h->Draw();`
    - `h.Draw();`
- Look at
  - http://root.cern.ch/root/html400/ClassIndex.html

Scenario 1 (Plain text file)

- Example from Linear Collider
  - `cd ~`
  - `mkdir root-intro`
  - `cp /unix/lc/sboogert/teaching/grad-root-2004/lumi_b0.dat ./root-intro/`
  - `cd ./root-intro`
  - Look at lumi_b0.dat
- 9 Columns of numbers
- Examples of exercises
  - Read this file in and make a histogram of column 1
  - Plot column 3 against 4

Scenario 1 (Plain text file)

- Solution (or what you should have learned)
  - Read in text file
    - `std::ifstream ifstr(fileName);`
    - `while(ifstr >> x1 >> x2) {}`
  - Making the histogram
    - `TH1F *x1 = new TH1D("x1","x1",100,240,260);`
    - `x1->Draw();`
  - Saving histogram file
    - `TFile *file = new TFile("s1.root", "recreate");`
    - `file->Write();`

ROOT files

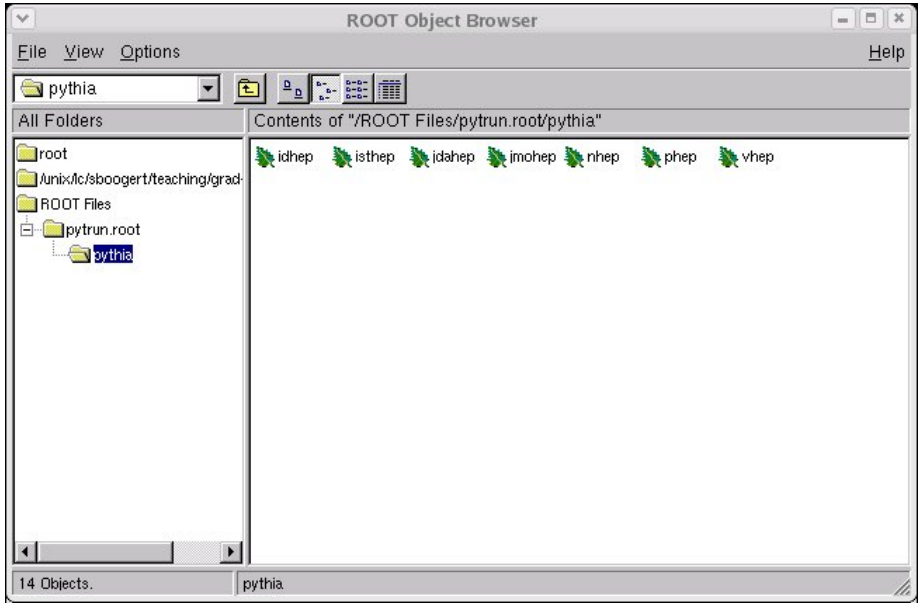
- Usually text files are not sufficient
  - Special file/data format like .jpg, .mp3, .doc, .xls .nb
- ROOT has a special file format
  - Format to store almost any type of data including
    - Histograms
    - Trees
  - Can store almost anything, well any type of object
    - TGraph
    - TLorentzVector
    - Objects you can/will define
      - e.g. NemoTrack
      - e.g. AtlasMuon
  - ROOT files have an internal directory structure like UNIX file system

What are ntuple/trees?

- Structure to store large volumes of numerical data
  - Each event stored as entry or event
  - Each entry can store single numbers, arrays of numbers or lists (variable size)
  - Excellent at storing a large volume of data which is of a similar type
    - Structure of event is the same for each event, contains
      - Tracks
      - Verticies
      - Jets
  - For example
    - Each event contains an array of particle momenta
      - px[npart], py[npart], pz[npart]
    - This structure is repeated for all the events

Scenario 2 (ROOT file)

- Given a ROOT file with little/no description, have to make plots and do analysis
  - Copy data file `[] cp /unix/lc/sboogert/teaching/grad-root-2004/pytrun.root ./root-intro/`
  - Start up root with: `[] root ./pytrun.root`
  - Start up a TBrowser: `root[] new TBrowser();`
- Click on ROOT files on left
  - Click on pytrun.root on left
    - Leaves on right



Scenario 2 (ROOT file)

- Solution
  - Extracting the tree
    - `TFile *f = new TFile("pytrun.root");`
    - `TTree *t = (TTree*)f->Get("pytrun");`
    - `t->MakeClass("pytrunAnalysis");`
  - Edit pytrunAnalysis.C Loop function
    - To include selection for electron positron pair
      - `if(idehep[7] == 11 && idhep[8] == -11) {`
    - Make TLorentzVectors of particles
      - `TLorentzVector v1 = TLorentzVector();`
      - `TLorentzVector v2 = TLorentzVector();`
    - Fill histogram with invariant mass
      - `h->Fill((v1+v2).Mag())`
      - `h->Draw();`

Scenario 3 (hbook file)

- Hbook is old package from CERN some experiments still use it
  - It is possible to convert hbook files to root files with the command
    - `[]h2root file.hbook file.root`
  - This will convert all the histograms and ntuples to ROOT histograms and TTrees
  - Example from ZEUS, where a ZEUS output file has been converted to ROOT TTree format
  - `[]cp /unix/lc/sboogert/teaching/grad-root-2004/file1.root ./root-intro`
  - Take a look at in a browser, make some histos!

ROOT startup configuration

- Rootlogon.C loaded when root starts
  - Write your own rootlogon.C file to load some of the macros you have written today with
    - `gROOT->LoadMacro("macrofile.C","k");`
  - Also possible to compile macros
    - `gSystem->CompileMacro("macrofile.C","k");`
  - Possible to set global apparence of plots and objects via gStyle
    - `gStyle->SetOptStat(kFALSE);`
    - `gROOT->ForceStyle();`
- ROOT uses global pointers for this
  - `gROOT`
  - `gStyle`
  - `gMinuit`
  - More on these in later lecture

Summary

- Learned how to
  - Load in ASCII file
  - Make histograms
  - Make code from a TTree file
  - Loop over the entries in a TTree and make histogram
  - Saving histograms to disk
  - Convert hbook files to root files
  - Introduction to the TBrowser, and TCanvas

Future

- Might not happen, depends on YOUR requirements

- Compiled ROOT programs
  - Write a full program which does not require interactive ROOT
  - Compile and link programs for analysis of large quantities of data
- More advanced fitting and analysis
  - Multidimensional histograms
  - Functions (TF1/2)
  - Fitting (TMinuit)
- More complicated IO
  - TTree branches and leaves (when I've worked out how to use them)
  - Managing output histograms/files
  - Analysis of multiple files
  - Writing your own objects to disk!