ROOT

An Object-Oriented Data Analysis Framework
Overview

- ROOT Executive Summary
- What ROOT can do for you: Examples
- Getting ROOT
- How to avoid this course
- What to expect in this course
The ROOT system provides a set of **OO frameworks** with all the functionality needed to handle and analyse large amounts of data in a very efficient way. Having the data defined as a set of **objects**, specialised storage methods are used to get direct access to the separate attributes of the selected objects, without having to touch the bulk of the data. Included are **histogramming methods in 1, 2 and 3 dimensions**, **curve fitting**, **function evaluation**, **minimization**, **graphics** and **visualization** classes to allow the easy setup of an **analysis system** that can query and process the data interactively or in batch mode.
Thanks to the built-in CINT C++ interpreter the command language, the scripting, or macro, language and the programming language are all C++. The interpreter allows for fast prototyping of the macros since it removes the time consuming compile/link cycle. It also provides a good environment to learn C++. If more performance is needed the interactively developed macros can be compiled using a C++ compiler.

You’ll be able to use ROOT without knowing C++, since many things are click-and-play ...
The system has been designed in such a way that it can query its databases in parallel on MPP machines or on clusters of workstations or high-end PC's. ROOT is an open system that can be dynamically extended by linking external libraries. This makes ROOT a premier platform on which to build data acquisition, simulation and data analysis systems.
Getting ROOT

ROOT can be freely downloaded from http://root.cern.ch

Binaries are available for many platforms:

- Intel x86 Linux for Redhat FedoraCore2 and gcc 3.3.3
- Intel x86 Linux for Redhat 9.0 and gcc 3.2.2
- Intel x86 Linux for Redhat RHEL 3 (SLC3) and gcc 3.2.3
- Intel x86 Linux for Redhat 7.3 and gcc 3.2.2
- Intel x86 Linux for Redhat 7.3 and gcc 2.96
- Intel x86 Linux for Redhat 7.3 and gcc 2.95.3
- Intel x86 Linux for Redhat 7.3 and gcc 2.95.2
- Intel x86 Linux for Redhat 9 and Intel’s icc 8.0
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and gcc2.95.2
- AMD x86 64 (Athlon64 and Opteron) FC 2 and gcc 3.4
- Intel Itanium Linux for RHEL3 and gcc 3.2.3
- Intel Itanium Linux for RHEL3 and Intel’s icc 8.0
- HP PA-RISC HP-UX 10.20 with aCC (v1.18)
- Compaq Alpha OSF1 with cxx 6.2
- Compaq Alpha OSF1 with cxx 6.2
- IBM AIX 4.5 with xC version 5
- Sun SPARC Solaris 5.7 with CC5.2
- Sun SPARC Solaris 5.8 with CC5.2
- SGI IRIX 6.5 with CC
- SGI IRIX 6.5 with g++ 2.95.2
- SGI IRIX 6.5 with KCC
- MacOS X 10.3.6 and gcc 3.3
- WindowsXP/NT/w2000 with CYGWIN and gcc3.3 version 4.03/02
- WindowsXP/NT/w2000 with VC++ 7.1 (runs with VC++6)
- Windows/NT/w2000 with VC++ 7.0 (runs with VC++6)

For all others: source code
How to avoid this course

ROOT has excellent documentation available online:

- Installation instructions: Very detailed
- User’s Guide: 470 pages
- Reference Guide: each class; also older versions
- Tutorials: 87 worked out examples
- HOWTO’s: 34 chapters
- RootTalk Forum: 1123 members, 5787 articles
- RootTalk Digest: all articles back to 1997; ROOT version 0.9
- Example Applications: 56 applications
- BaBar Tutorials: not so clear
- FNAL Tutorials: C++, presentations, exercises, examples
- MINOS Tutorials: somewhat specific; excellent for new C++'ers
What to expect in this Course

We’ll clearly not cover everything = 1058159 lines of code in 918 classes (a little under 47 MB)

LEARNING GOALS:

1. Knowledge and experience to do basic analysis from the command line
2. Ability to convert private data format to ROOT data structures
3. Skills to do more complex analysis
4. Make publication-worthy plots

REQUIREMENTS:

1. Access to computer with ROOT (version 4.03/02)
2. Some programming experience (in whatever language)
3. Data to play with

SCHEDULE: once a week for 1 hour till about June (tuesdays, 13:00–14:00)?

MATERIAL: ROOT used guide, FNAL course and local development
The END
Frameworks
Some words on ‘Object Oriented’

Hiding complex things behind something simple ....
Histogramming
Fitting

data courtesy Umakanth Dammalapati
Functions

Abs ACos ACosH ASin ASinH ATan ATan2 ATanH Bessel BesselI0 BesselI1 BesselJ0 BesselJ1 BesselK BesselK0 BesselK1 BesselY0 BesselY1 Beta BetaCf BetaDist BetaDistI BetaIncomplete BinarySearch Binomial BreitWigner BubbleHigh BubbleLow C CauchyDist Ccgs Ceil Cos CosH Cross CUncertainty DegToRad DiLog E Erf Erfc ErfcInverse ErfInverse Even Exp Factorial FDist FDistI Finite Floor Freq G Gamma GammaDist Gaus Gcgs GeomMean GhbarC GhbarCUncertainty Gn GnUncertainty GUncertainty H Hash Hash Hbar HbarCgs HbarUncertainty HC HCcgs Hcgs HUncertainty Hypot InvPi IsInside IsNaN K Kcgs KolmogorovProb KOrdStat KUncertainty Landau LaplaceDist LaplaceDistI Ldexp Ln10 LnGamma LocMax LocMin Log Log10 Log2 LogE LogNormal Max MaxElement Mean Median Min MinElement MWair Na NaUncertainty NextPrime Nint Nint Normal2Plane Normalize NormCross NormQuantile Odd Permute Pi PiOver2 PiOver4 Poisson Power Prob Qe QeUncertainty R RadToDeg Range Rgair RMS RootsCubic RUncertainty Sigma SigmaUncertainty Sign Sin SinH Sort Sqrt StruveH0 StruveH1 StruveL0 StruveL1 Student StudentI StudentQuantile Tan TanH TwoPi Voigt

and of course any other function that you might want to program yourself (code or symbolically)
Minimization

Full implementation of MINUIT:

- squared differences, $\chi^2$, and $\log L$ standard
- something else also possible
- fitting with bounded parameters
- fitting over sub-range of data
- full evaluation of errors and correlations
- and much more ...

Alternative:

Neural networks
Graphics
**EDM ring @ KVI**

- $E = 100$ MeV
- $p = 625$ MeV/c
- $E = 6$ MV/m
- $B = 0.44$ T
- $R = 12$ m
- $L_q = 40$ cm
- $L_d = 150$ cm
Analysis System
Differences from PAW

- Regular grammar (C++) on command line
- Single language (compiled and interpreted)
- Object Oriented (use your class in the interpreter)
- Advanced Interactive User Interface
- Well Documented code. HTML class descriptions for every class.
- Object I/O including Schema Evolution
- 3-d interfaces with OpenGL and X3D.