PROOF tutorial
ROOT Basics

Gerardo Ganis, CERN, PH-SFT
gerardo.ganis@cern.ch

How to get ROOT

- Using pre-build binaries from
  - http://root.cern.ch
  - CERN AFS:
    - /afs/cern.ch/sw/lcg/app/releases/ROOT/
  - ATLAS CVMFS:
    - /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/x86_64/root
- Building the binaries from SVN source
  - Check
    http://root.cern.ch/drupal/content/installing-root-source
  - Check also the pre-requisites
    http://root.cern.ch/drupal/content/build-prerequisites

External dependencies: XROOTD

- Many, but one, XROOTD, particularly important for PROOF … and data access in general
- XROOTD is external since ROOT v5.32
  - Before it was delivered with ROOT
- Now available from
  - CERN AFS and ATLAS CVMFS
    - /afs/cern.ch/sw/lcg/external/xrootd/
    - /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/x86_64/xrootd
  - Binaries from http://xrootd.slac.stanford.edu/
  - From source: $ROOTSYS/build/unix/installXrootd.sh

Setting up ROOT

- The installation copies the relevant files under the chosen path /path/to/root
- To setup the environment (ROOTSYS, …) just do

```
$ source /path/to/root/bin/thisroot.sh
```

- Remember to setup also XROOTD

```
$ source /path/to/root/bin/setxrd.sh /path/to/xrootd
```

with /path/to/xrootd the XROOTD installation path

Trying out

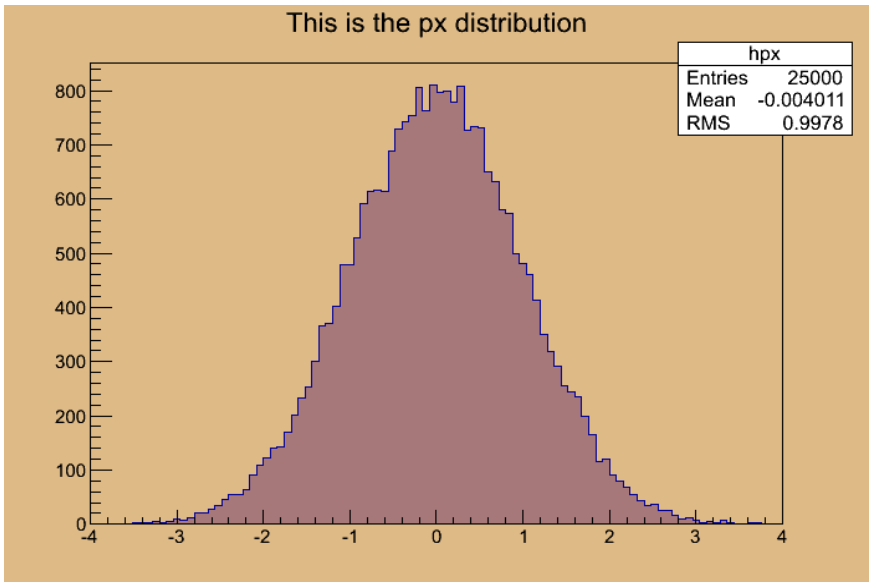
- Copy the ROOT tutorials directory to your area

```
$ cp -rp $ROOTSYS/tutorials .
```

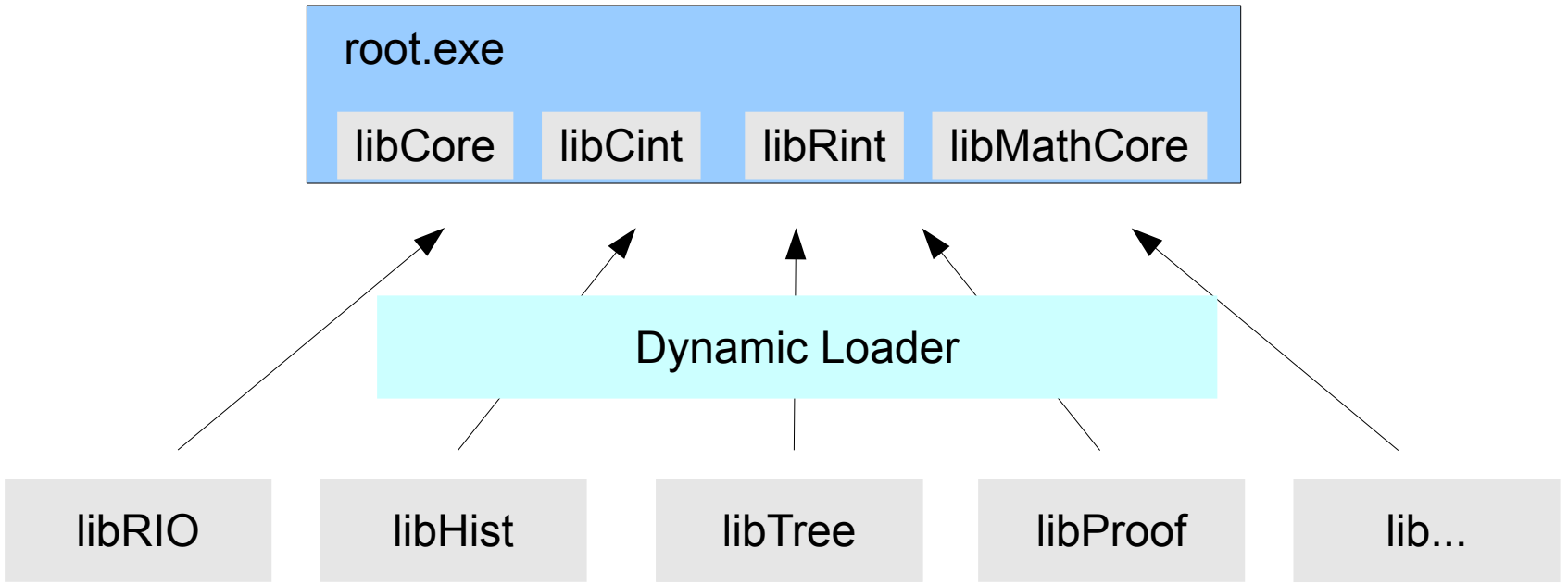
- Run the demos.C macro from the copied dir

```
$ cd tutorials
$ root -l
root[] .x demos.C
```

- Click on the 'hsimple' button to get the 'Dynamic Filling Example' histogram

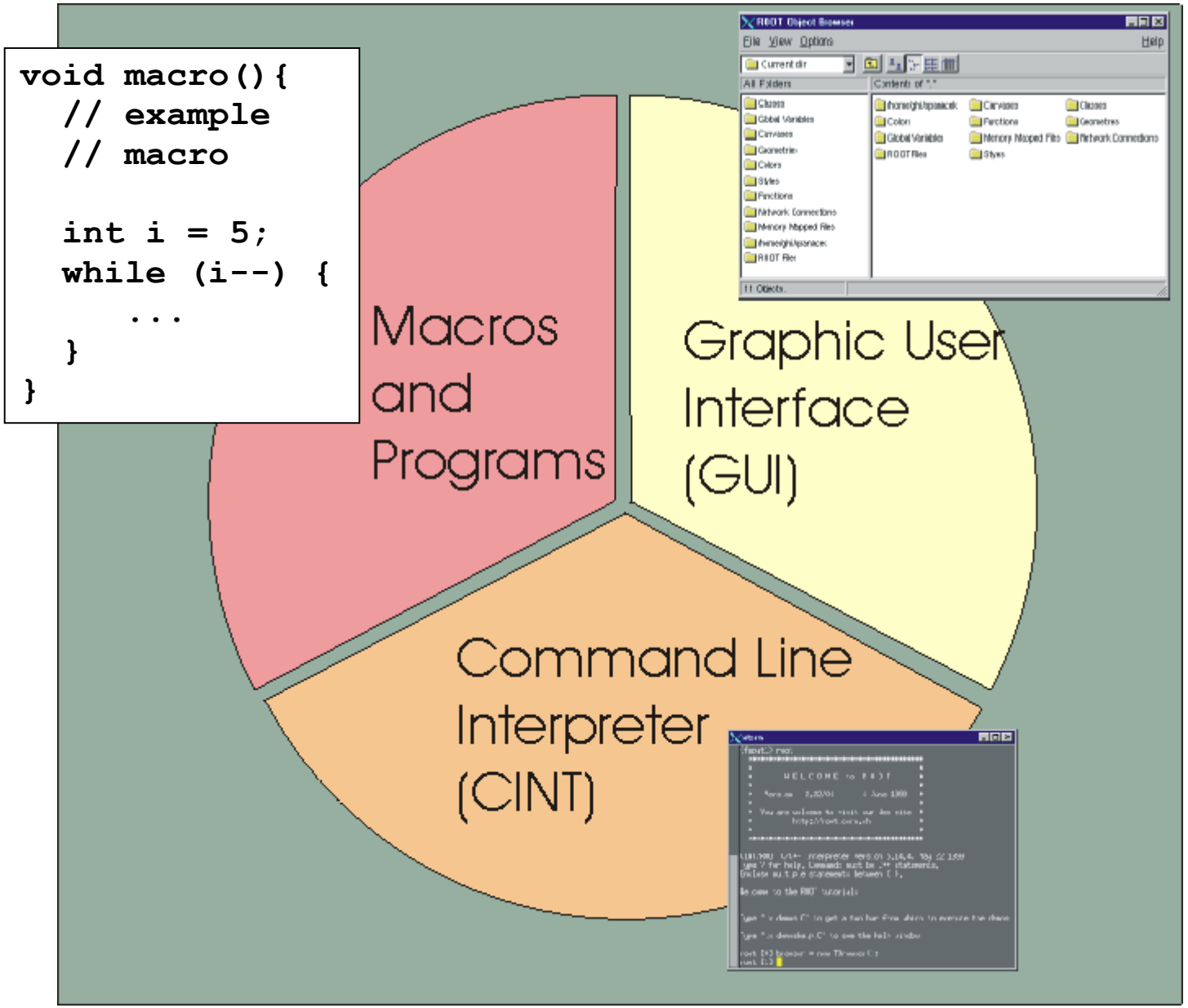


ROOT Layered Library Structure



- Core libs: support for RTTI, basic I/O, interpreter
- Shared libs (plug-in) loaded on demand
  - Optimize link time and application size
  - Dependencies reduced
- libXXX.rootmap tell the loader what is needed

Three User Interfaces



Shell and Macros

- ROOT shell

```
root [] _ // Any C++ or C command (almost ...)
```

- Unnamed macros
  - Bunch of commands to be executed in the shell
    - Everything in the global scope

```
{ // global scope
   int f = 1, n = 5;
   while (n > 1) {
      int j = 1;
      f *= n--;
   }
}
```

- Named macros
  - (Multiple) – functions to be called by name
    - C++ scope respected

```
int fact (int n) {
   int f = 1;
   while (n > 1) {
      int j = 1;
      f *= n--;
   }
   return f;
}
```

CINT interpreter

- The prompt runs the CINT interpreter
  - Can interpret also scripts
- CINT (C/c++ INTterpreter) understands C and C++
- Covers most of ANSI C and ISO C++ …

`http://root.cern.ch/viewvc/trunk/cint/doc/limitati.txt`

- The functions srqt(), sin(), cos(), printf(), ... are part of the C library and are built-in
- CINT offers a gdb-like debugging environment
- CINT knows class internals and can generate dictionaries required for I/O

Some CINT commands

- Probably most used
  - .x, .X *macro.C* — execute macro.C
  - .L file — load content of file
  - .q — exit ROOT
  - .h — Display available commands
- Macro debugging (à la gdb)
  - .T — Tracing on/off
  - .f *macro.C* — Set macro to debug
  - .b *line* — Set break at line
  - .S *macro()* — Debug macro()
  - .p *var* — Examine variable var

Running some code

To run the C function test_trig() in file test_trig.c:

```
root [0] .x exercises/c/test_trig.c(.34)
```

Equivalent: load file and run function:

```
root [0] .L exercises/c/test_trig.c
root [1] test_trig(.34)
```

Compiling the code: ACLiC

- Interpreting code is convenient, but slow: compiling the code into a shared library speeds-up things
- ACLiC (Automatic Compiler of Libraries for CINT)
  - Just add a '+' at the end

```
root [0].x exercises/c/test_trig.c+(.34)
```

  - This creates and loads exercises/c/test_trig_c.so
  - Now CINT knows all the functions / types in the library:

```
root [1] test_trig(.68)
```

- NB: functions must C/C++ compliant
- Use '++' to force re-compilation

Running macros in batch mode

- root –b –q myscript.C
  - myscript.C is executed by the interpreter
- root –b –q “myscript.C(42)”
  - same as above, but giving an argument
- root –b –q “myscript.C+”
  - myscript.C is compiled via ACLIC with the native compiler and linked with the executable.
- root –b –q “myscript.C++g(42)”
  - same as above. Script compiled in debug mode and an argument given.

ROOT portable basic data types

Signed	Unsigned	sizeof [bytes]
`Char_t`	`UChar_t`	1
`Short_t`	`UShort_t`	2
`Int_t`	`UInt_t`	4
`Long64_t`	`ULong64_t`	8
`Float_t`		4
`Double_t`		8
`Double32_t`		float on disk, double in RAM

ROOT classes and TObject

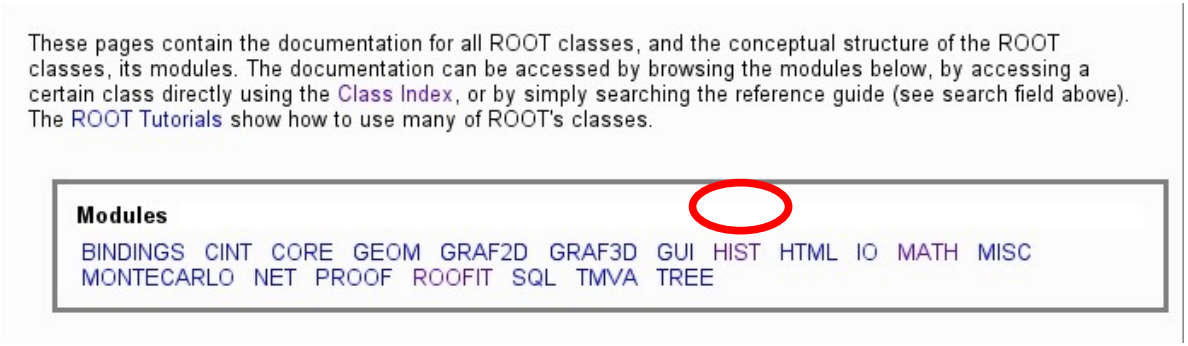
- All ROOT classes names start with T (for Type)
- TObject
  - lightweight class defining default behavior and protocol for the objects in the system
- It provides
  - Reflection and Run-Time-Type-Identificaion (RTTI)
  - Interface with the I/O system
  - Clone functionality
  - A 32-bit mask very convenient to store 1-bit info
- Deriving from TObject not mandatory
  - But allows to fully exploit the system functionality of the system
    - E.g. schema evolution, collection classes

TNamed

- TNamed is like TObject with two additional members of type TString (built-in string class):
  - fName, the name of the object
  - fTitle, the title of the object
- Many ROOT classes derive from TNamed:
  - Search by name very common in ROOT
  - Leaving one of the two empty adds very little to the class
  - No change in the interface as TObject has default implementations of GetName() and GetTitle()

ROOT class categories

The top of http://root.cern.ch/root/html gives



Each module correspond to a category and to a given functionality

E.g. HIST contains

i.e. the basics to work with histograms and functions

Typically only a few classes are used directly, e.g. TH1F, TH2D

The others are base or auxilliary classes used internally

Dump class w/ CINT

- Use .class ClassName to dump the content

```
root [] .class TH1F
================================================================================
class TH1F //1-Dim histograms (one float per channel)
 size=0x3d0 FILE:/home/ganis/local/root/xrdhead/root/lib/libHist.so LINE:-1
 (tagnum=444,voffset=-1,isabstract=0,parent=-1,gcomp=0:-1,funcs(dn21=~xcpd)=ff)
List of base class-----------------------------------------------------------
0x0        public: TH1 //1-Dim histogram base class
  0x0        public: TNamed //The basis for a named object (name, title)
    0x0        public: TObject //Basic ROOT object
  0x40       public: TAttLine //Line attributes
  0x50       public: TAttFill //Fill area attributes
  0x60       public: TAttMarker //Marker attributes
0x3b8      public: TArrayF //Array of floats
  0x0        public: TArray //Abstract array base class
List of member variable------------------------------------------------------
Defined in TH1F
/home/ganis/local/root/xrdhead/root/lib/libHist.so  -1 0x0        private: static TClass* fgIsA
List of member function------------------------------------------------------
filename       line:size busy function type and name  (in TH1F)
libHist.so  -1:-1   0 public: virtual void ~TH1F(void);
libHist.so  -1:-1   0 public: TH1F TH1F(void);
libHist.so  -1:-1   0 public: TH1F TH1F(const char* name,const char* title,Int_t nbinsx, ...);
libHist.so  -1:-1   0 public: TH1F TH1F(const char* name,const char* title,Int_t nbinsx, ...);
libHist.so  -1:-1   0 public: TH1F TH1F(const char* name,const char* title,Int_t nbinsx, ...);
libHist.so  -1:-1   0 public: TH1F TH1F(const TVectorF& v);
libHist.so  -1:-1   0 public: TH1F TH1F(const TH1F& h1f);
libHist.so  -1:-1   0 public: virtual void AddBinContent(Int_t bin);
libHist.so  -1:-1   0 public: virtual void AddBinContent(Int_t bin,Double_t w);
libHist.so  -1:-1   0 public: virtual void Copy(TObject& hnew) const;
libHist.so  -1:-1   0 public: virtual TH1* DrawCopy(Option_t* option="") const;
libHist.so  -1:-1   0 public: virtual Double_t GetBinContent(Int_t bin) const;
libHist.so  -1:-1   0 public: virtual Double_t GetBinContent(Int_t bin,Int_t) const;
libHist.so  -1:-1   0 public: virtual Double_t GetBinContent(Int_t bin,Int_t,Int_t) const;
libHist.so  -1:-1   0 public: virtual void Reset(Option_t* option="");
libHist.so  -1:-1   0 public: virtual void SetBinContent(Int_t bin,Double_t content);
...
```

Classes are auto-loaded

The classes provided by ROOT are known to the prompt, even thought not loaded when you start the application

The relevant rootmap file gives the information, so no need to load the library

For example, TH1F (1D histo of floats) is in libHist.so, not loaded at startup

```
root [0] strstr(gSystem->GetLibraries(),"libHist")
(const char* 0x0)""
root [1] h1 = new TH1F()
(class TH1F*)0xacff50
root [2] strstr(gSystem->GetLibraries(),"libHist")
(const char* 0x9d4188)"libHist.so ...”
```

Check $ROOTSYS/lib/libHist.rootmap for TH1F

gROOT is a singleton

- Singleton instance of TROOT, provides list of classes, canvases, files, …
  - See http://root.cern.ch/root/html/TROOT.html
- And some other useful functionality; e.g.
  - ProcessLine() to process 'prompt-like' lines in macros

```
// Load the macro to get data sets
gROOT->ProcessLine(".L getDataSet.C+");
```

  - Time() to measure CPU, real time of the last action

```
root [0] gROOT->Time(); gROOT
(class TROOT*)0xb7eed940
Real time 0:00:00.001455, CP time 0.010
```

gSystem is another singleton

- TSystem, provides OS abstraction layer
  - See http://root.cern.ch/root/html/TSystem.html
- Some examples
  - Load(*lib*) to load a plugin

```
root [] gSystem->Load("libPhysics")
```

  - HostName() to get the hostname

```
root [] gSystem->HostName()
(const char* 0x101218a20)"macphsft12.local"
```

  - BaseName(), DirName()

```
root [] gSystem->DirName("/path/to/file)
(const char* 0x1013035a0)"/path/to"
root [] gSystem->BaseName("/path/to/file")
(const char* 0x104c1cc91)"file"
```

TSystem (cnt'd)

- TSystem provides also access to file information
  - AccessPathName to test existence, … of a file

```
root [0] gSystem->AccessPathName("hsimple.C")
(Bool_t)0  <<== File exists
```

  - GetPathInfo to get all information about a file

```
root [] FileStat_t fst
root [] gSystem->GetPathInfo("hsimple.C", fst)
(int)0
root [] fst.fSize
(Long64_t)3641
```

- Transparent access to remote back-ends

```
gSystem->AccessPathName("root://dataserver//path/hsimple.C")
```

File on a xrootd server

Output re-direction w/ gSystem

- Often is useful to collect the out into a file
- In CINT

```
root [] .> out.txt
Output will be saved in file out.txt!
root [] cout << “This goes” << endl
root [] Printf(“Not this one”);
root [] .>
```

  - NB: Acts on 'cout << “Some output” << endl' ONLY!
- TSystem provides RedirectOutput

```
root [] gSystem->RedirectOutput(“out.txt”, “w”)
root [] cout << “This goes” << endl
root [] Printf(“Also this”)
root [] gSystem->RedirectOutput(0)
```

  - Works on all output streams
  - But it redirects also the shell prompt, so use it in macros

gEnv is yet another singleton

- TEnv, provides flexible dedicated environment
  - See http://root.cern.ch/root/html/TEnv.html
- Variables are *{key,value}* pairs in the form

```
<Scope>.<name>:  <value>
```

- Specified via file
  - In the order: rootrc, $HOME/.rootrc, system.rootrc; or the explicitly passed file

```
# Use old zib library
Root.ZipMode 0
```

- Or run time

```
root [] gEnv->SetValue("Root.ZipMode", 0)
```

- Check active values with **`gEnv->Print()`**

TString

- Class to handle strings, i.e. “this is a string”
- More features than std::string
  - See http://root.cern.ch/root/html/TString.html
- TString is not a TObject
  - Use TObjString if TObject functionality needed
  - See http://root.cern.ch/root/html/TObjString.html
- Example of simple string

```
root [] TString s("a=")
root [] s += 5
(class TString)"a=5"
root [] s[2]
(char 53)'5'
root [] s.IsDigit()
(const Bool_t)0
```

TString: more examples

```
// Make it a digit
root [] s.ReplaceAll("a=","")
(class TString)"5"
root [] s.IsDigit()
(const Bool_t)1

// Get the integer value
root [] s.Atoi()
(const Int_t)5

// Format it
root [] s.Form("This is %s string w/ a %d and a %f", "a", 5, 3.14)
root [] s
(class TString)"This is a string w/ a 5 and a 3.140000"

// Tokenization with regular expression
root [] s = “A string, right?”
root [] TString t; Ssiz_t from = 0;
root [] s.Tokenize(t, from, "[, ]"); Printf("t: '%s'", t.Data());
t: 'A'
root [5] s.Tokenize(t, from, "[, ]"); Printf("t: '%s'", t.Data());
t: 'string'
root [6] s.Tokenize(t, from, "[, ]"); Printf("t: '%s'", t.Data());
t: 'right?'
```

Collection classes

- ROOT collection classes are polymorphic
- They work on TObject derivations
  - E.g. TList

```
root [0] TList aList
root [1] aList.Add(new TNamed(“red”, “a color”))
// Casting needed when retrieving the object
root [2] TNamed *nm = (TNamed *) aList.FindObject(“red”)
```

  - Casting required when accessing objects
- Many others THashList, TSortedList, TMap, TObjArray, ...
  - See the http://root.cern.ch/root/html/CORE_CONT_Index.html

Example: iterating on TList

- All collection classes contain a MakeIterator
- It allows to easily generate a TIter instance

```
// Create the iterator
root [3] TIter nextentry(&aList)
// Iterate
root [4] TNamed *nm = (TNamed *) nextentry()
// Restart from beginning
root [5] nextentry.Reset()
```

Printf, Info, Warning, Error

- C++ provides cout << … << endl for printouts
- C-style technology (printf, fprintf, …) may be more concise for advanced formatting
- ROOT provides these functions (TError.h)
  - Printf: like 'printf' but no need for final '\n'
    - Everywhere (included on prompt):
  - Info, Warning, Error
    - Inside classes automatically adds header in front of formatted message, e.g.

```
Info in <TProofBench::SetOutFile>: using default ...
```

Class name

Method name

Converters and Examples

- Example

```
root [0] Int_t i = 5; Long64_t ll = 34;
root [1] Printf("i = %d, ll = %lld", i, ll);
i = 5, ll = 34
root [0] Float_t f = 3.14; Double_t d = 1.234;
root [1] Printf("f = %f, d = %f", f, d);
f = 3.140000, d = 1.234000
root [2] const char *cc = "const char *"; TString s("a string");
root [3] Printf("cc = '%s', s = '%s'", cc, s.Data());
cc = 'const char *', s = 'a string'
```

- Most useful converters (see 'man fprintf')
  - '%d' for 4 byte integers
  - '%f' for floats and doubles
  - '%s' for strings
- Lot of possibilities for advanced formatting
  - '%t' for tabs, '%r' for repositioning, etc ...

Customizing printouts

- The function which in the end does the printout can be overloaded for customization
- PROOF does does:

```
01:10:18  1175 Wrk-0.1 | Info in <TProofServ::Setup>: fWorkDir: /tmp
```

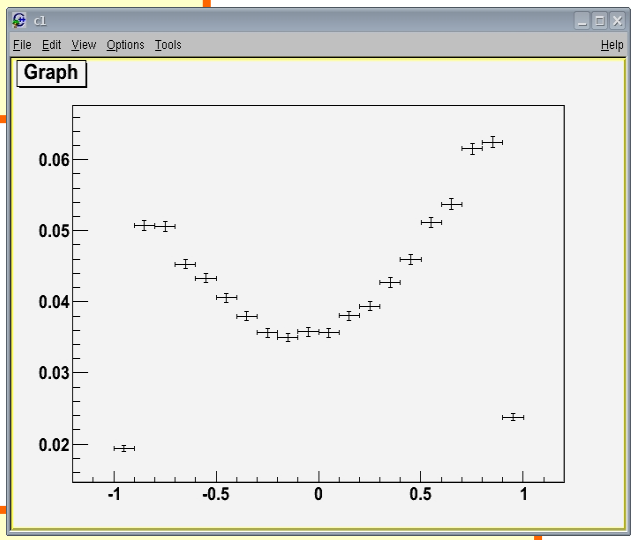
- The relevant calls:

```
#include “TError.h”

void MyErrorHandler(Int_t level, Bool_t abort,
                    const char *location, const char *msg)
{
   // My own ErrorHandler function
   ...
}

    // Changing the log leveles:
    //    abort on higher than kSysError's and set error handler
    gErrorAbortLevel = kSysError + 1;

    // Using MyErrorHandler
    SetErrorHandler(MyErrorHandler);
```

Example using TGraphErrors (from HIST)

- A list of measurements in a text file files/afb.txt

```
 -0.950000 0.019410 0.050000 0.000441
 -0.850000 0.050760 0.050000 0.000712
 -0.750000 0.050660 0.050000 0.000712
...
```

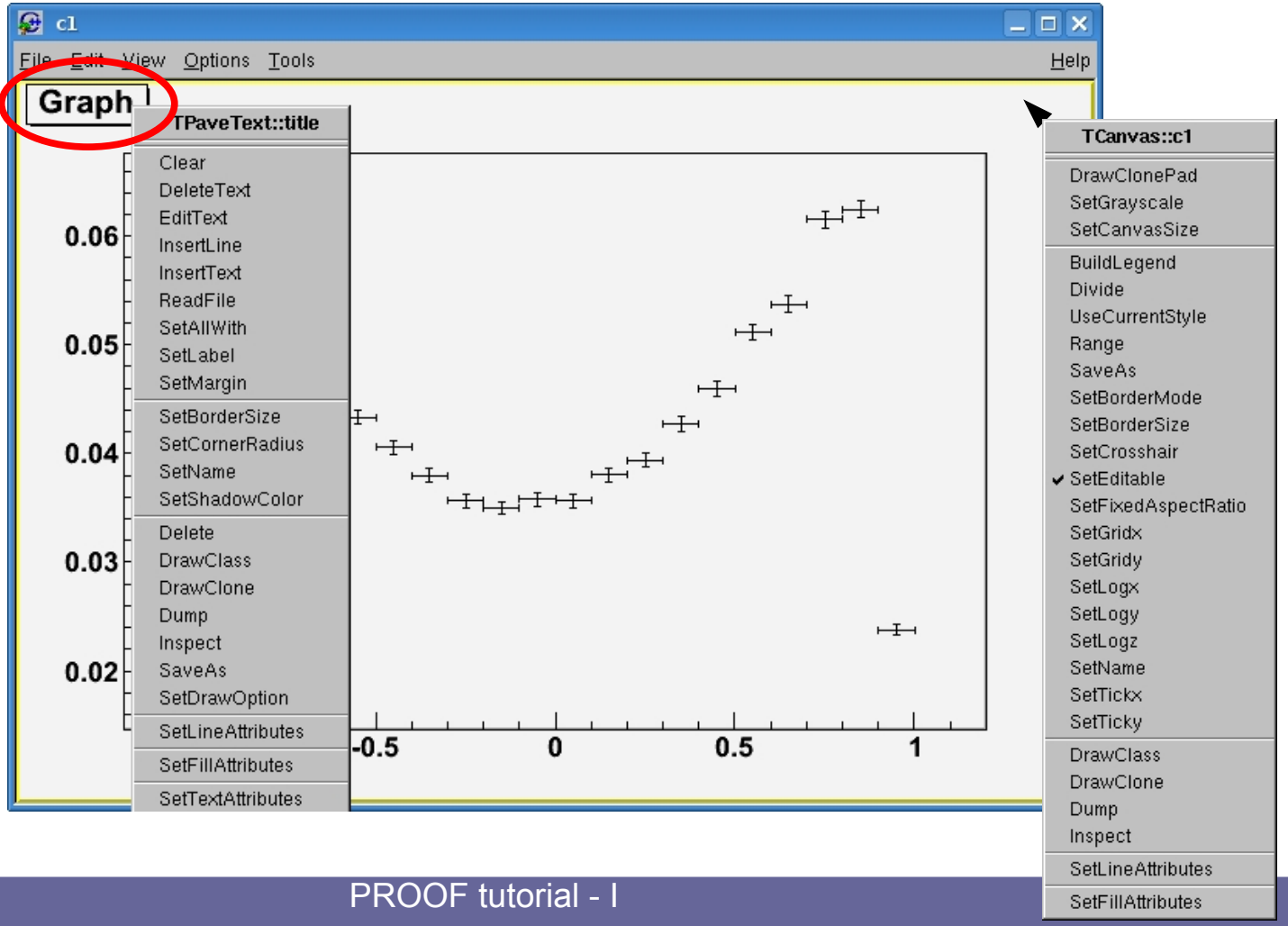


- Visualization helps understanding
  - TGraphErrors

```
root[] TString afbdata("files/afb.txt")
root[] TGraphErrors *g = new TGraphErrors(afbdata)
// Display by drawing axes and polymarker, if any
root[] g->Draw("AP")
```

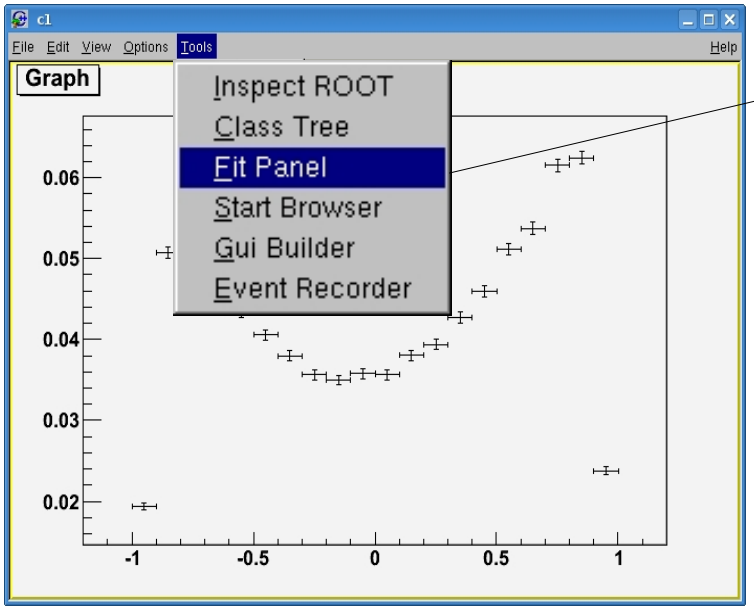
TCanvas : the graphic output

- The output is displayed into an instance of TCanvas
- All displayed components are objects: right-click on them to find out the originating class



Example of using the fit panel

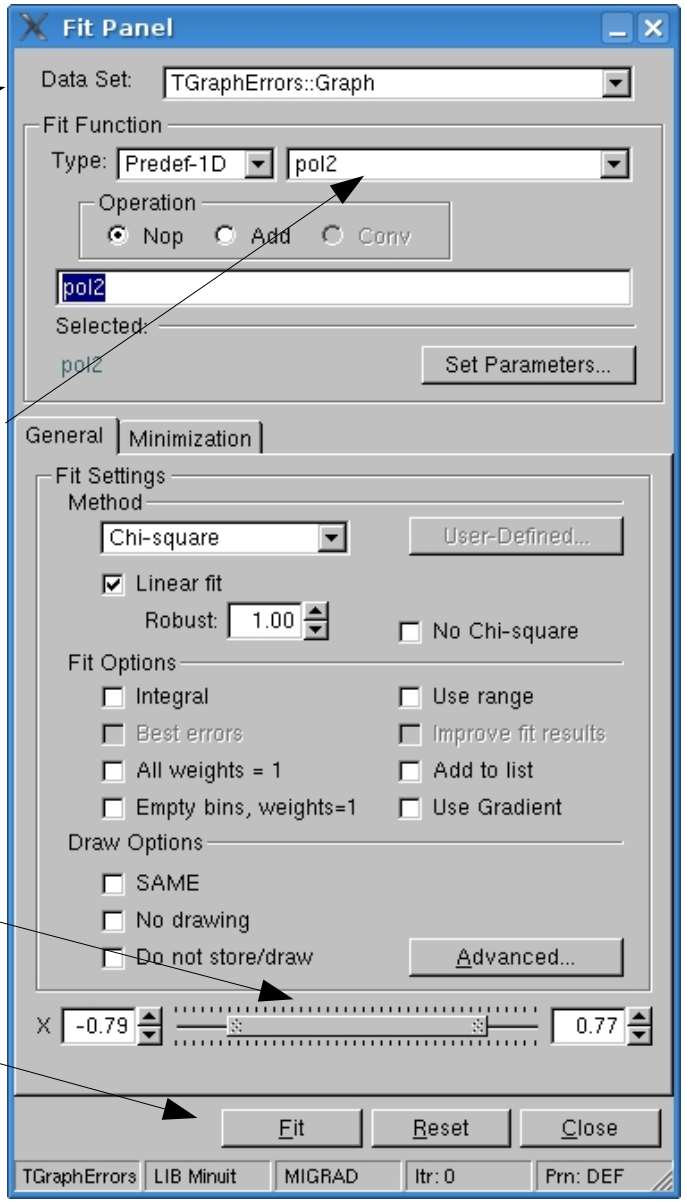
- Open the fit panel from the tools tab



Set the function

Set the range

Run the fit



- Where is the result of the fit?

Example of fitting from the command line

- You can do the same from the command line

```
// Show result of the fit
root[] gStyle->SetOptFit(1011);
// Define the fitting funtion
root[] TF1 *fb = new TF1("fb","[0]*(1+[1]*x+x*x)",-.8,.8);
// Set the initial parameters
root[] fb->SetParameters(1.,.2);
// Run the fit in the range specified by the function
// super-imposing the result
root[] g->Fit(fb,"R","S")
```

- You can export the graphical result of the fit into the standard graphic formats from the File menu of the TCanvas object

Remember

What are TSystem, TNamed, TList, TGraphError?

What functions do they have?

Documentation!

User's Guide, Tutorials, HowTo's:

http://root.cern.ch

Reference Guide (full class documentation):

http://root.cern.ch/root/html