Lecture 2

The basic concepts behind an Object-Oriented framework

Exercise 1

Find out which method changes the markerstyle of a 1D histogram.

Inheritance tree of a 1D histogram (*e.g.* a TH1D):

class TH1D : public TH1, public TArrayD

class TH1 : public TNamed, public TAttLine, public TAttFill, public TAttMarker

Class TAttMarker:

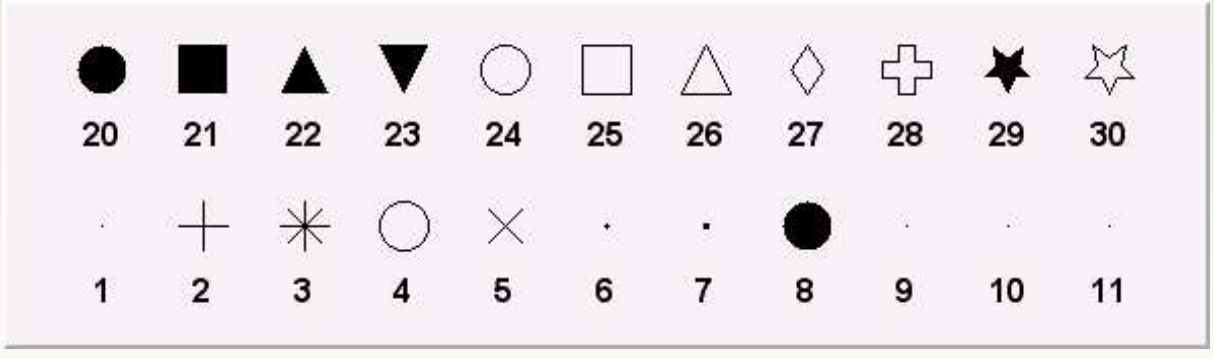
```
*-*-*-*-*-*-*-*-*-*-*-*Marker Attributes class*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-*                    ======================
*-*  Marker attributes are:
*-*    Marker Color
*-*    Marker style
*-*    Marker Size
*-*
*-*  This class is used (in general by secondary inheritance)
*-*  by many other classes (graphics, histograms).
...
```

Answer: virtual void SetMarkerStyle(Style_t mstyle = 1)

Example: myHistoPtr->SetMarkerStyle(kFullSquare)

Exercise 1: Marker styles

```
*-*  List of the currently supported markers (screen and PostScript)
*-*  ================================================================
*-*      1 : dot                     kDot
*-*      2 : +                       kPlus
*-*      3 : *                       kStar
*-*      4 : o                       kCircle
*-*      5 : x                       kMultiply
*-*      6 : small scalable dot      kFullDotSmall
*-*      7 : medium scalable dot     kFullDotMedium
*-*      8 : large scalable dot      kFullDotLarge
*-*      9 -->15 : dot
*-*     16 : open triangle down      kOpenTriangleDown
*-*     18 : full cross              kFullCross
*-*     20 : full circle             kFullCircle
*-*     21 : full square             kFullSquare
*-*     22 : full triangle up        kFullTriangleUp
*-*     23 : full triangle down      kFullTriangleDown
*-*     24 : open circle             kOpenCircle
*-*     25 : open square             kOpenSquare
*-*     26 : open triangle up        kOpenTriangleUp
*-*     27 : open diamond            kOpenDiamond
*-*     28 : open cross              kOpenCross
*-*     29 : open star               kOpenStar
*-*     30 : full star               kFullStar
```



Exercise 2

Create a 2D histogram and fill it with a gaussian distribution.

Slightly modify Johan's example:

```
void doit()
{
  TH2F *myHis = new TH2F("myHis","A Histogram Example",100,-1.,1.,100,-1.,1.);
  TRandom *ranNumGen = new TRandom();
  TCanvas *myCanvas = new TCanvas("myCanvas","This is a drawing table",1);
  myCanvas->SetFillColor(kWhite);
  myCanvas->SetFrameFillColor(kYellow);
  myHis->SetFillColor(kBlue);
  myHis->SetLineColor(kBlue);
  for (Int_t i=0; i<100000; i++)
  {
    myHis->Fill(ranNumGen->Gaus(0.5,0.2),ranNumGen->Gaus(0.5,0.2));
    if ((i%200)==0)
    {
      myHis->Draw();
      // myCanvas->Refresh(); <=== TYPO
      myCanvas->Update();
    }
  }
}
```

Exercise 3

Create a canvas and put a text with your name in it. Specify which classes are used and which methods and print the results as a postscript file.

From the Root tutorials on
http://root.cern.ch/root/html/examples/hello.html:

```
Root >  TPaveLabel hello(0.2,0.4,0.8,0.6,"Gerco says Hello!");
Root >  hello.Draw();
Root >  c1->Print("hello.eps");
```

Exercise 4

Find the longest chain of pointers possible using the TH1 as base class, I.e. object->a()->b()->c()...

Many possibilities .....

```
h->Rebin(1)->Rebin(1)->Rebin(1).....

h->GetDirectory()->FindObject("h")->.....

h->GetFunction("pol1")->GetExpFormula()->Contains("pol")
```

Lecture 3

Finding your way in ROOT memory:
Names, Lists, Directories, Browsers and Files

This Lecture in Short

As soon as you start to have a significant analysis session using ROOT, keeping track of all the objects you created may become difficult.

To make things easier, objects are usually created with a name, which can be used to look up the object.

Compare this with you (the Object), your phone number (pointer to the Object) and the phone book which links you and your phone number via your name.

Objects with Names

Nearly every object inherits from TNamed, *i.e.* it has a *name*. For example, the TH1 class definition reads:

```
class TH1 : public TNamed, public TAttLine, public TAttFill, public TAttMarker
```

and in the constructor you find:

```
TH1(const char* name, const char* title, Int_t nbinsx, Axis_t xlow, Axis_t xup)
```

This is how it is used:

```
TH1D* myHistoPtr = new TH1D("myHisto","Title of this histo",10,0,1);
```

Be careful not to use the same name twice .... or you'll get

```
Warning in <TH1::Build>: Replacing existing histogram: myHisto (Potential memory leak).
```

Listing objects

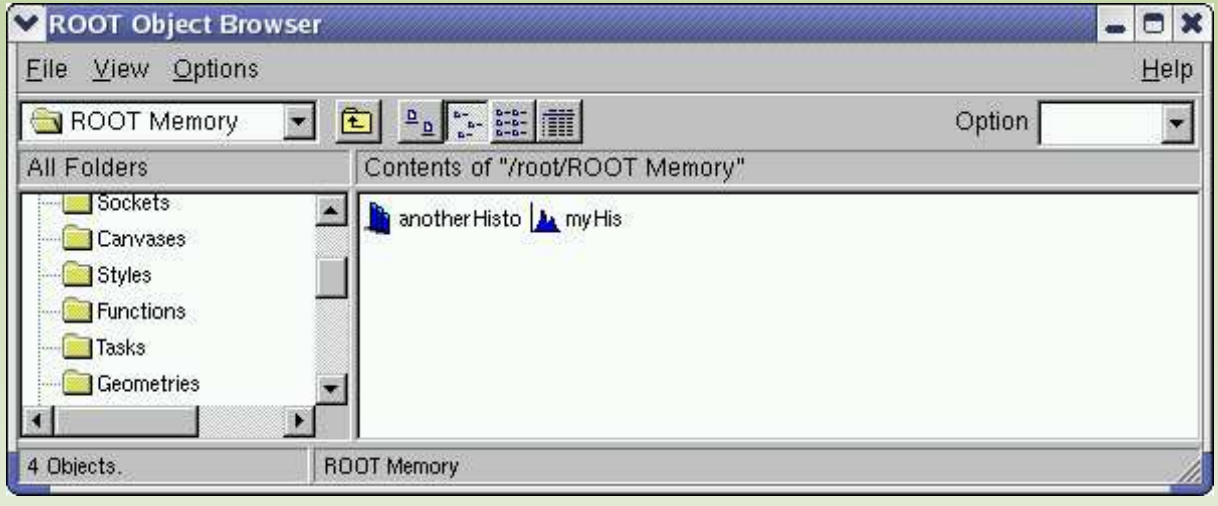
You can see which objects you created using the .ls command:

```
root [1] .x test.cxx
root [2] TH2D* h2 = new TH2D("anotherHisto","This is a 2D hist",10,0,1,10,0,1)
root [3] .ls
TROOT*          Rint    The ROOT of EVERYTHING
OBJ: TH1F       myHis        A Histogram Example : 0 at: 0x8c784c8
OBJ: TH2D       anotherHisto  This is a 2D hist   : 0 at: 0x8c865d0
```

This may become cumbersome if there are a lot of objects around ...

TBrowser

The alternative to .ls is the TBrowser class:

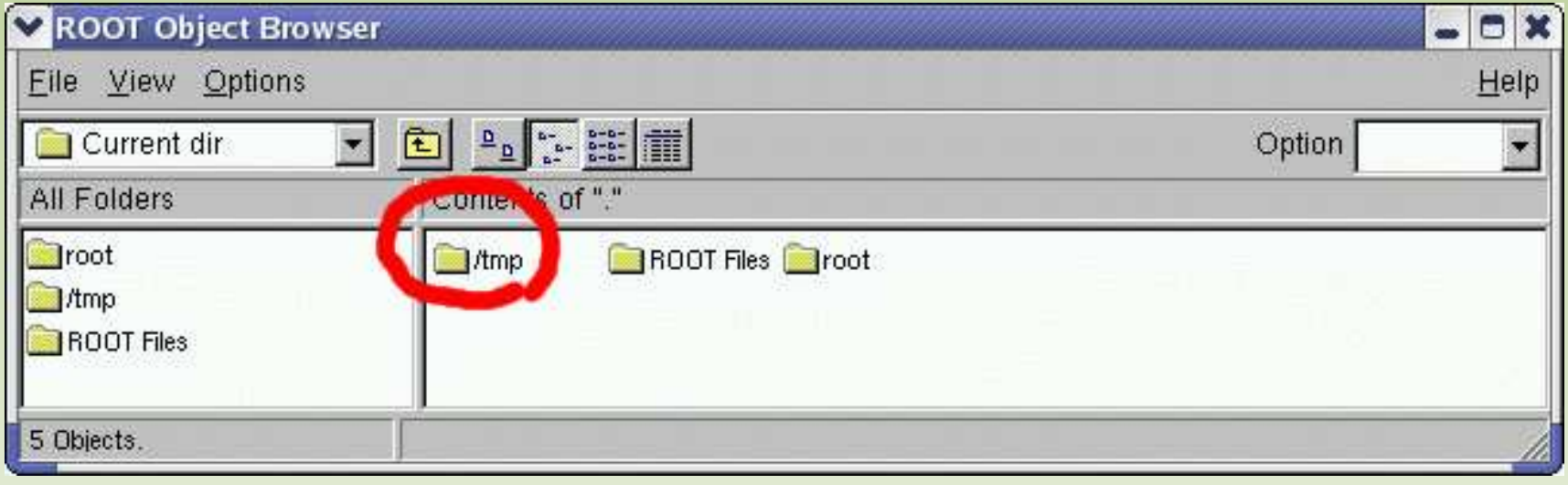


The browser is simply started with the command

```
root [1] new TBrowser
```

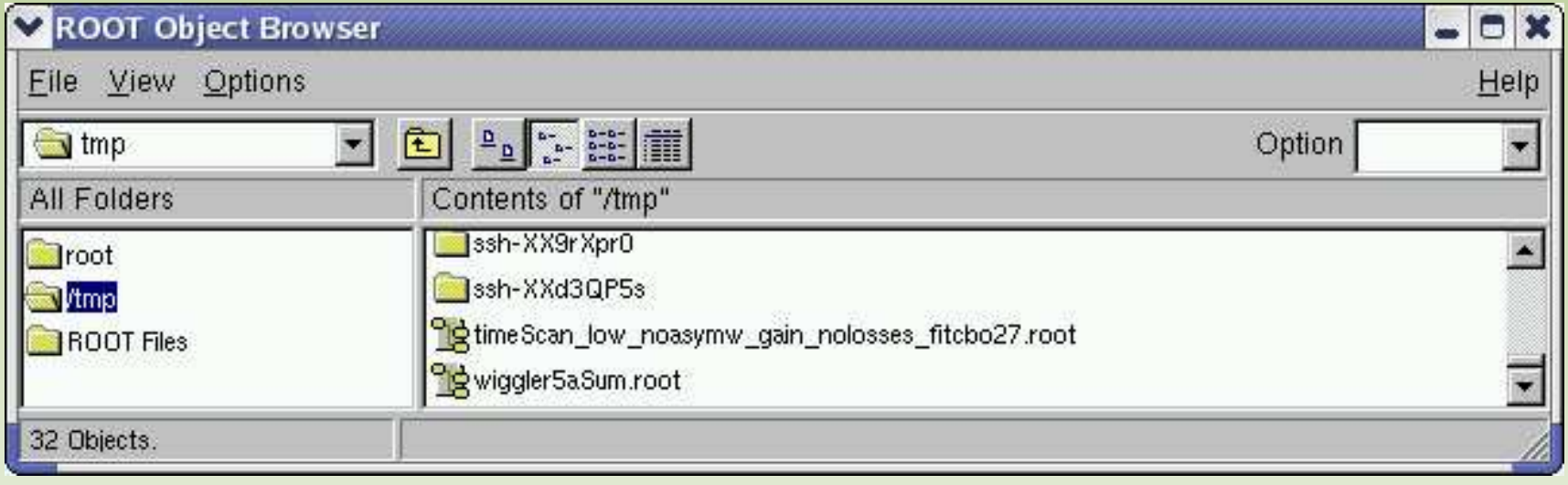
Browsing the Contents of a ROOT File

A file can be opened via the Browser



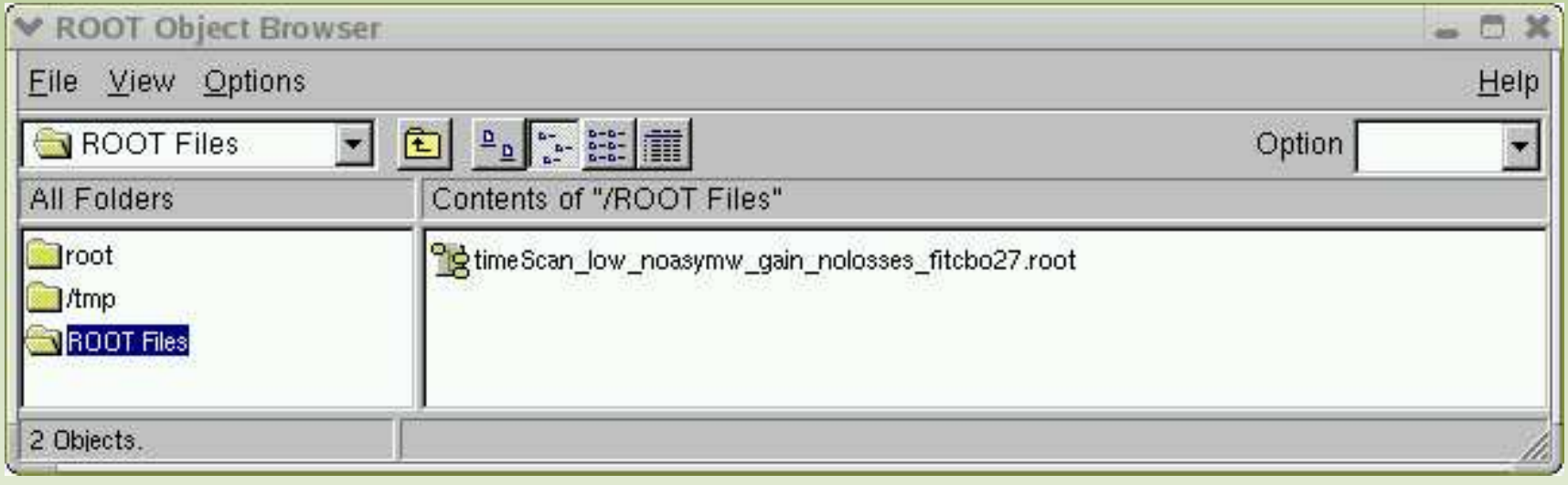
Browsing the Contents of a ROOT File

A file can be opened via the Browser



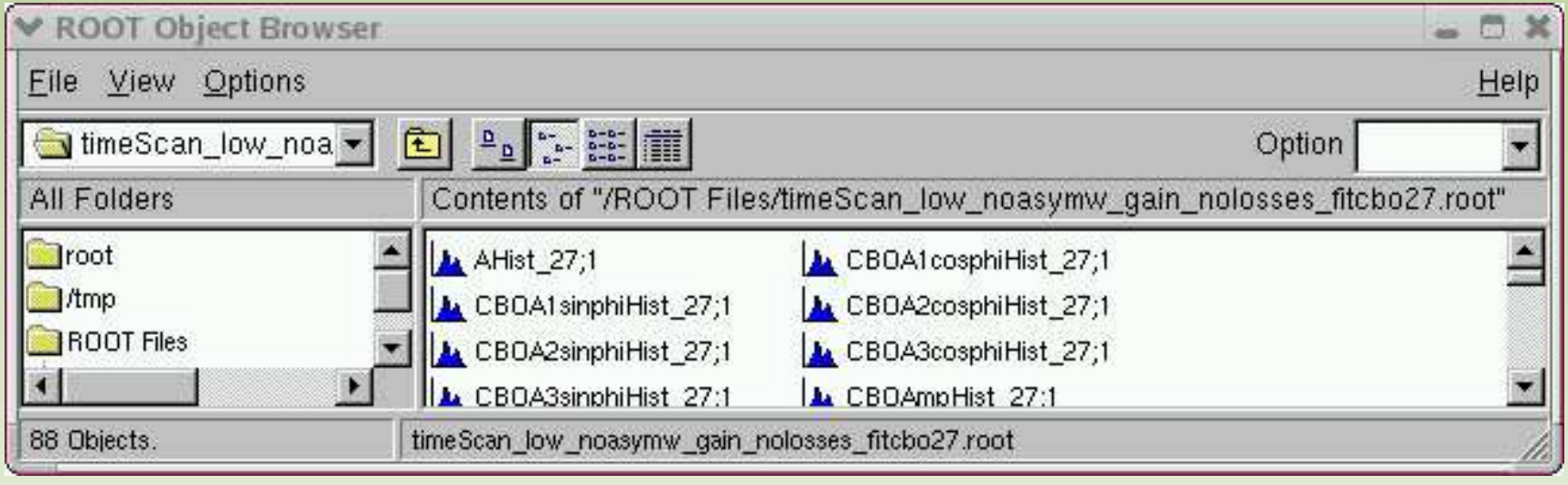
Browsing the Contents of a ROOT File

A file can be opened via the Browser



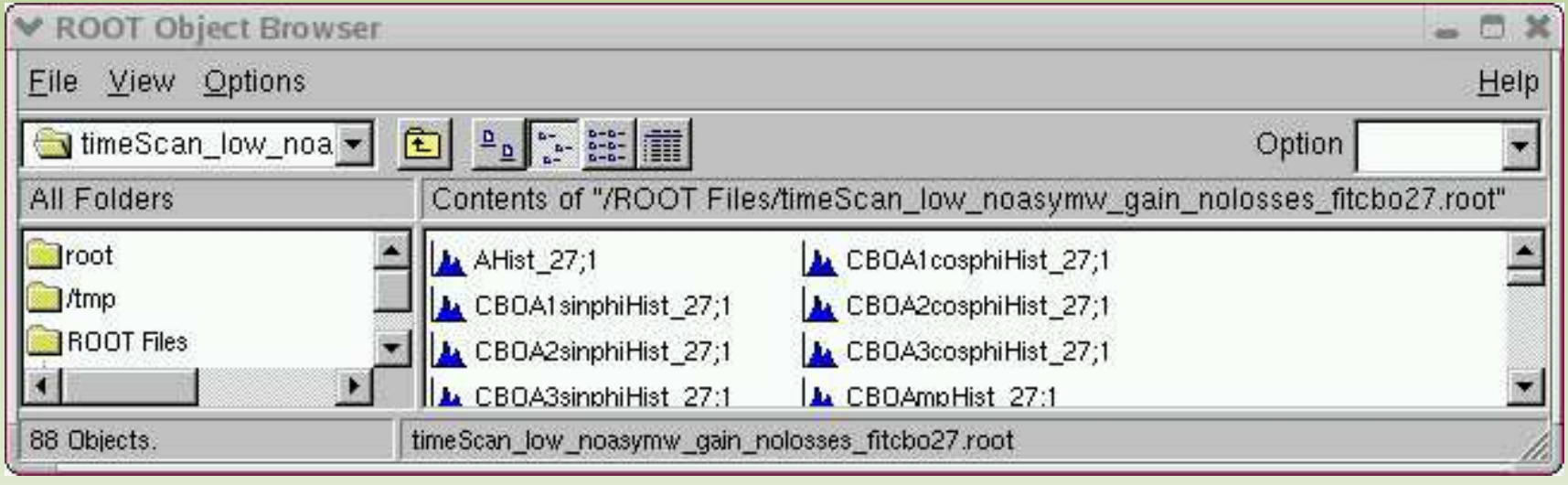
Browsing the Contents of a ROOT File

A file can be opened via the Browser



Browsing the Contents of a ROOT File

A file can be opened via the Browser



Alternative:

```
root [1] f = new TFile("/tmp/timeScan_low_noasymw_gain_nolosses_fi tcbo27.root")
```

Drawing a Histogram From a File

Option 1: in the browser, click on the histogram

Option 2: the proper commandline/macro version

```
root [1] TFile* f = new TFile("/tmp/timeScan_fitcbo27.root")
root [2] TH1D* h = (TH1D*)f->Get("fastRot24")
(class TH1D*)0x8d9fdd8
root [3] h->Draw()
```

Get uses the *name* to look up an object in the file, reads it into memory and returns a pointer to it.

Option 3: the sloppy version

```
root [1] TFile* f = new TFile("/tmp/timeScan_fitcbo27.root")
root [3] fastRot24->Draw()
```

Here, ROOT uses the *name* of the object to find it by itself

A Few Words on TFile

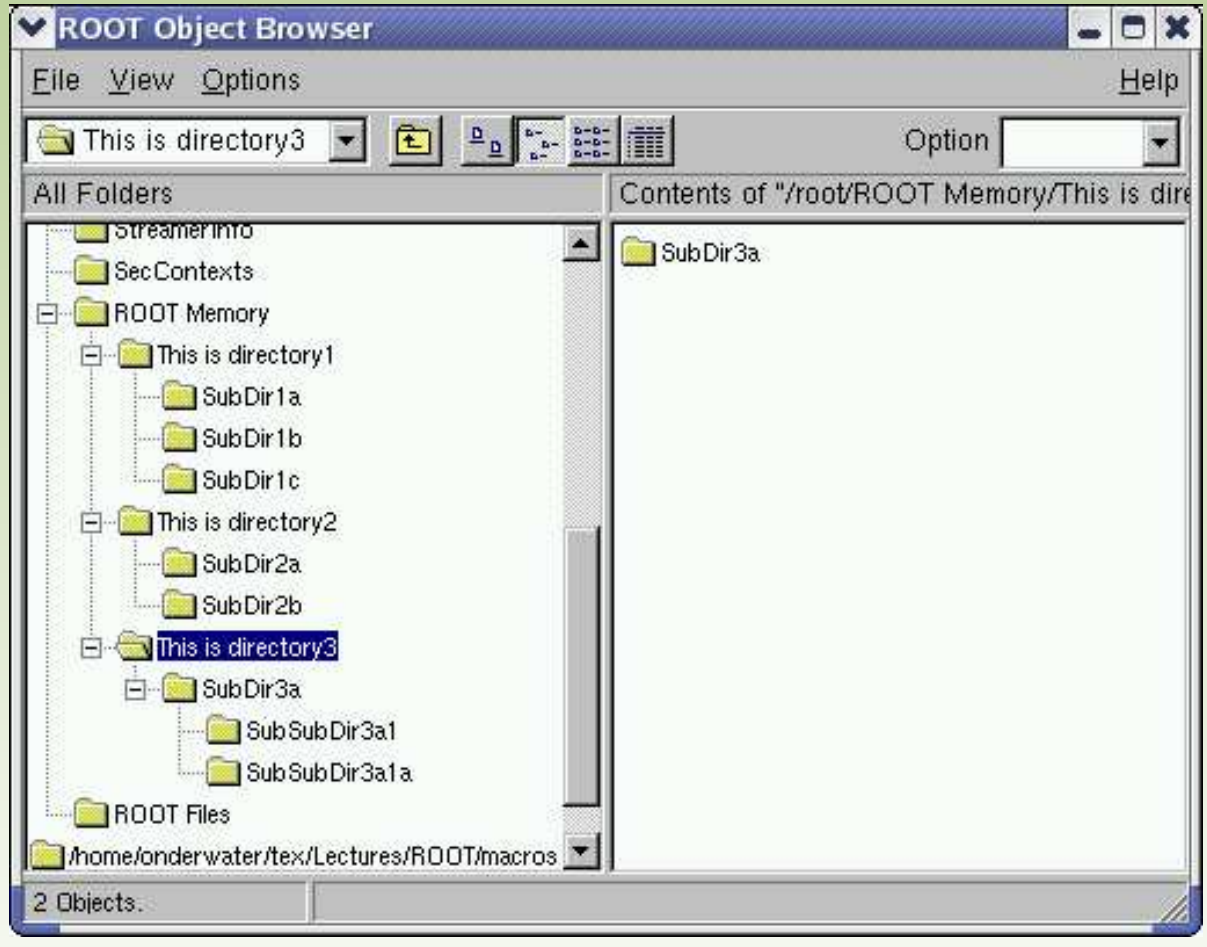
ROOT has its own IO interface, which is implemented in the class TFile

A ROOT file (like a Unix file system) may contain objects and directories. There are no restrictions for the number of levels of directories.

A ROOT file can be used interactively. In this case, one has the possibility to delete or change existing objects and add new ones. **An opened file behaves as any other directory in memory.**

A ROOT file is by default compressed, so that it uses a minimal amount of disk space

Directory Structure



Making New Directories

The top level directory is gROOT:

```
root [0] TDirectory* d1 = gROOT->mkdir("d1","This is directory1")
root [1] TDirectory* d2 = d1->mkdir("d2","This is subdir1")
root [2] TDirectory* d3 = gROOT->mkdir("d3","This is directory2")
root [3] .ls
TROOT*           Rint    The ROOT of EVERYTHING
 TDirectory*             d1      This is directory1
  TDirectory*            d2      This is subdir1
 TDirectory*             d3      This is directory2
root [4] d2->cd()
(Bool_t)1
root [5] .ls
TDirectory*             d2      This is subdir1
```

You can navigate through this directory structure with the browser or by hand

Global Variables

In addition to gROOT, there are some more global variables that apply to the session:

gROOT: gROOT holds information relative to the current session. By using the gROOT pointer you get the access to basically every object created in a ROOT program.

gFile: gFile is the pointer to the current opened fi le.

gDirectory: gDirectory is a pointer to the current directory.

gPad: A graphic object is always drawn on the active pad. It is convenient to access the active pad, no matter what it is. For that we have gPad that is always pointing to the active pad.

gRandom: gRandom is a pointer to the current random number generator.

gEnv: gEnv is the global variable with all the environment settings for the current session.

Exercises

① Use the browser to find out *which* standard presentation styles are available in ROOT. Hint: there are 5

② Read `http://root.cern.ch/root/htmldoc/TFile.html` and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there ....

③ Open the file you created in (2) in *update* mode and change the title of the histogram. Describe what you did.

④ From gEnv (an instance of the TEnv class), get the name of the default fitter in ROOT: "Root.Fitter". Hint: for *dflt* use ""