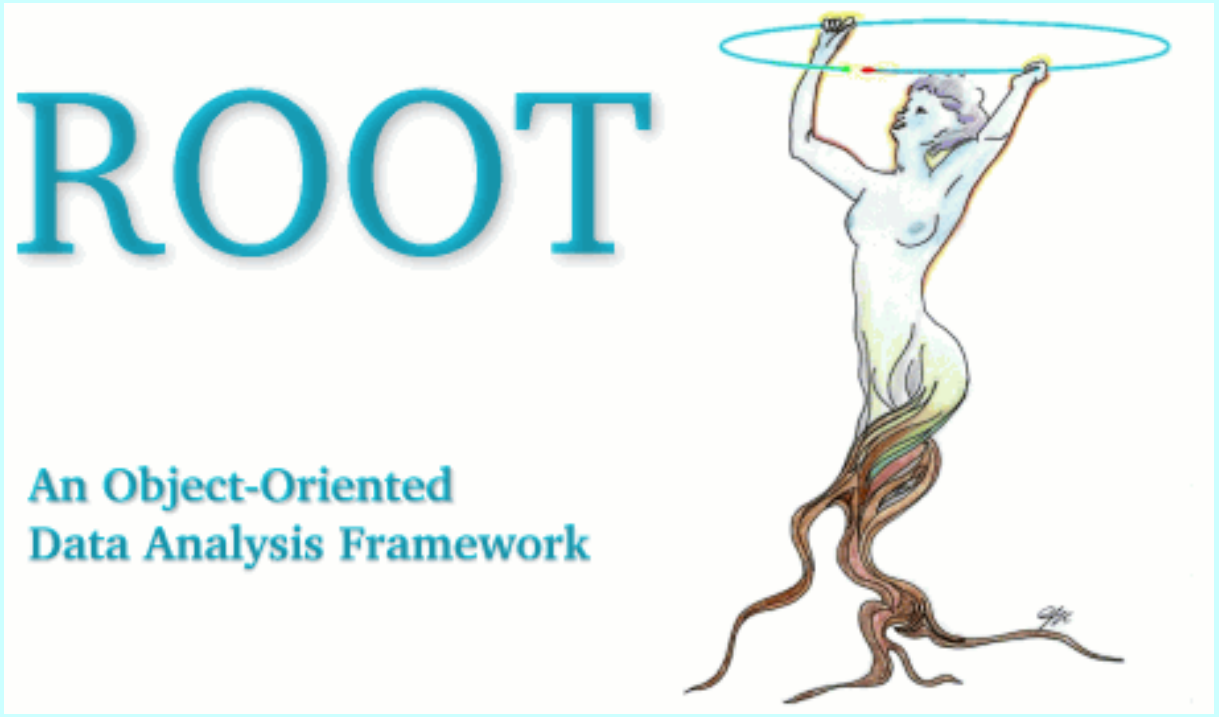
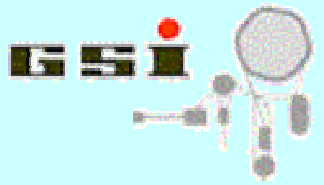


The ROOT framework 1: Introduction

J. Adamczewski

DVEE C++ course 2005



Schedule of part 1

- ROOT system overview
- The ROOT C++ interpreter CINT
- CINT and ROOT GUI for interactive work
- C++ as script language
- CINT runtime compiler ACLiC
- Compiled C++ with ROOT libraries
- (ROOT framework as class hierarchy)

The ROOT system

C++ based framework for (high energy-) physics

(developed at CERN by R.Brun et al. since 1997)

- **C++ as script language with interpreter CINT**
- **GUI for interactive visualization (TCanvas, TBrowser,...)**
- **I/O and analysis of large amount of data (TFile, TTree,...)**
- **Histograming, plotting, fits (TH1x, TGraph, TF1,...)**
- **Physics and mathematics (TMatrix, TLorentzVector, TMath,..)**
- **Object organisation (TCollection, TDirectory, TFolder,...)**
- **Parallel analysis via network (TProof)**
- **...**
- **see http://root.cern.ch for further info!**

Getting started (at GSI linux)

. rootlogin [version]

e.g. :

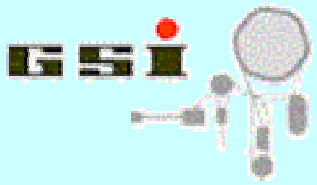
```
> . rootlogin 403-04
> . rootlogin dev
> . rootlogin
```

see http://root.gsi.de for available versions

The C++ interpreter CINT

- Written by Masa Goto in C
- Standalone (non root) version available
- Incorporated in ROOT
- Applied for:
  - Interactive command line
  - Macro execution (language: C++ with extras);
    may compile macro at runtime (ACLiC feature)
  - Generates object introspection metadata („dictionary“) when compiling ROOT

CINT command line



1. CINT native commands start with “.”

.?

list all the CINT commands

.x [filename]

load [filename] and execute function [filename]

e.g.: `root[1] .x plot.C(“go4asf”)`

.L [filename]

load [filename]

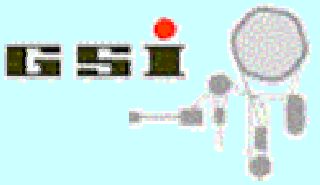
e.g. `root[2] .L plot.C`

.! [shellcmd]

execute shell command

e.g. `root[3] .! ls -al`

CINT command line (cont.)



2. **Expression evaluation (advanced calculator):**

```
root[3] 3*4
(int)12
```

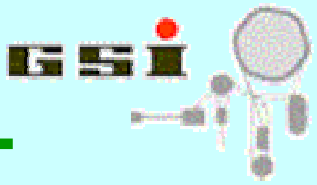
3. **C++ Syntax (almost; see below...):**

```
root [0] TBrowser *b = new TBrowser()
   or
root [0] TBrowser *b = new TBrowser();
```

Leave off final semicolon to see the return value of the command.

```
root [0] 23+5  // show return value
   (int)28
root [1] 23+5; // no return value
root [2]
```

CINT: differences to compiled C++



1. **Declaration can be omitted**

```
f = new TFile("Example.root");
```

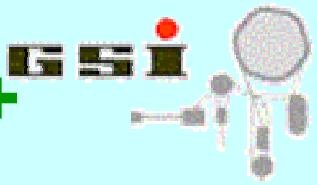
2. **Member access: "." and "->“ both possible**

```
f.ls() or f->ls()
```

3. **Local object is not deleted when leaving scope**

```
{
   TString s(“test”);
   cout << s.Data()<<endl; // OK
}
cout << s.Data()<< endl; // only in CINT!
```

CINT: differences to compiled C++ (cont.)



4. **Unknown variables are automatically initialized by searching the object of that name in gROOT.**

```
TH1F *smallHisto = new TH1F ("small","fPx”,100,100,-5,5);
small->Draw();
```

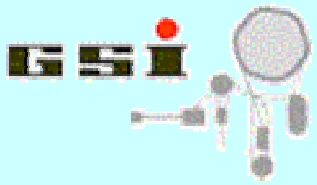
Implicitely the following is done (as correct for compiled code):

```
TH1F* small=dynamic_cast<TH1F*> (gROOT->FindObject("small"));
small->Draw();
```

C++ variable name

ROOT object name

Using ROOT classes interactively



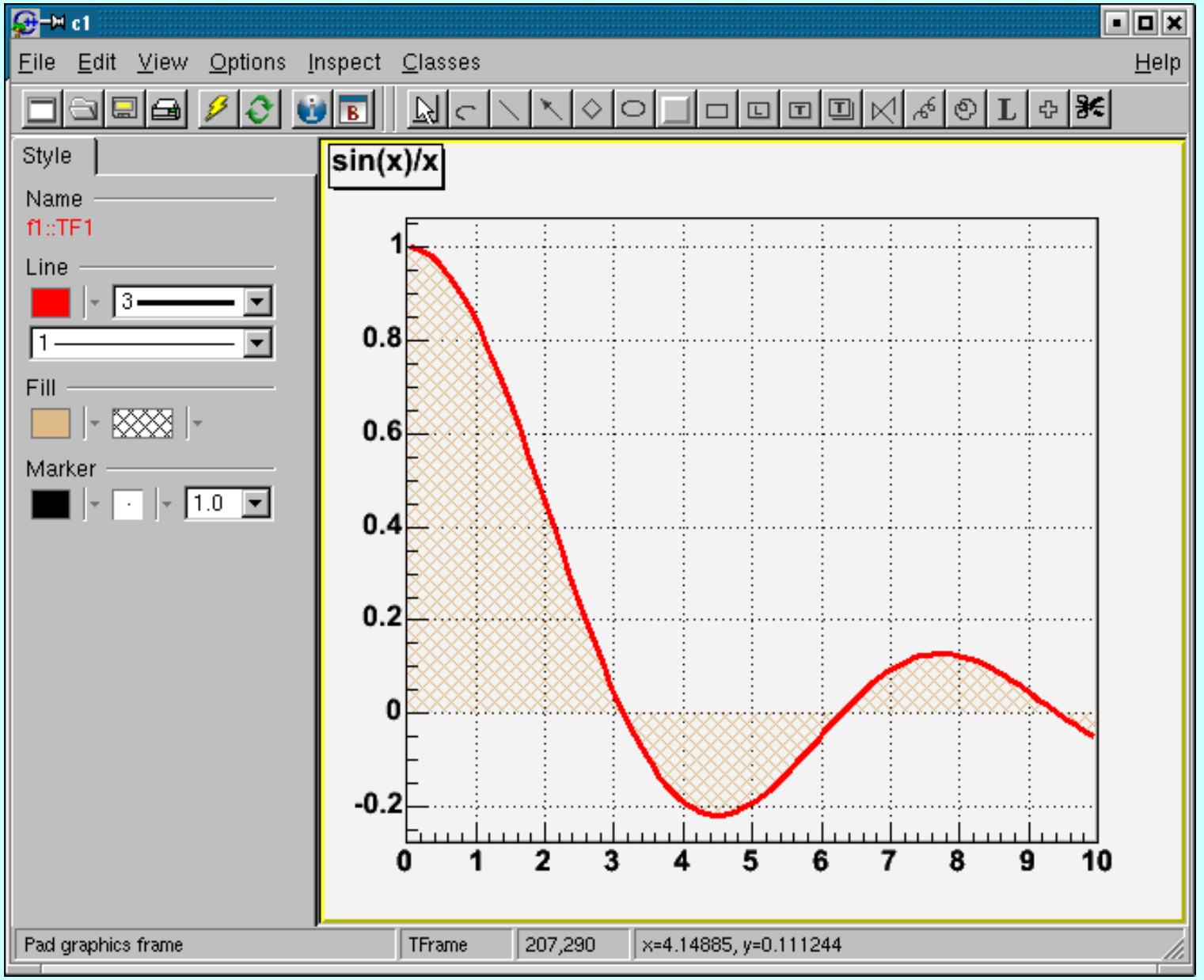

```
>root
root [] TF1 f1("function1","sin(x)/x",0,10);
root [] f1.Draw();          // pure C++
       or
root [] function1.Draw(); //cint reference by root name

root [] f1.Eval(1.2456789)
...
root [] f1.Derivative(2.3)
..
root [] f1.Integral(0,3)
...
root [] .q
```

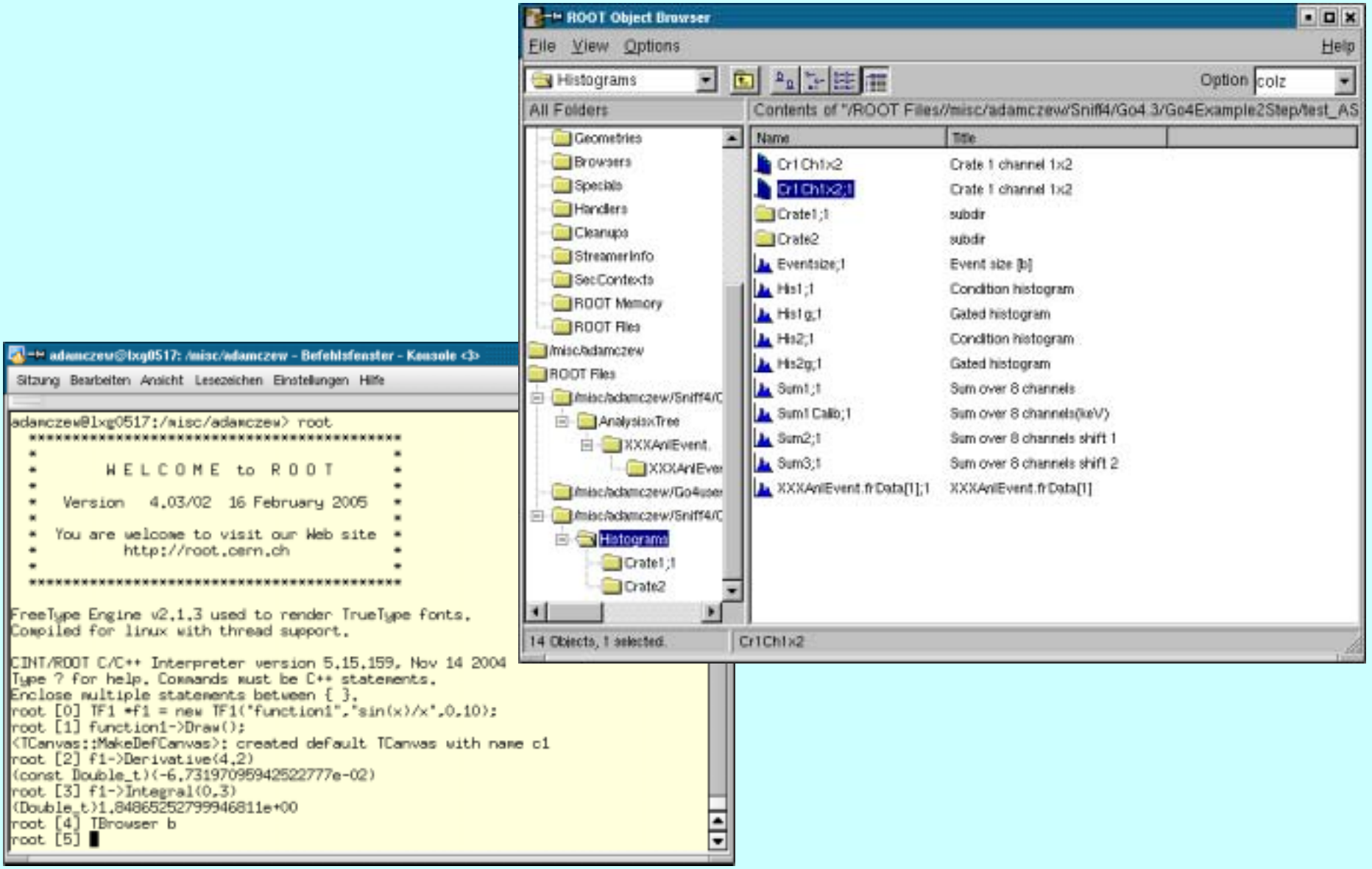
See ROOT class documentation for existing methods!

http://root.cern.ch/root/html/TF1.html

Root GUI: TCanvas



Root GUI: TBrowser



Global ROOT objects (some examples)

- gROOT (session object singleton, class TROOT)

```
TObject* ob=gROOT->FindObject("hpx");
   // get known object by root name
TSeqCollection* list= gROOT->GetListOfFiles();
   // access to collections of registered objects
gROOT->SetBatch(kTRUE);
 // switch to non graphic mode
```

- gSystem (operating system interface, class TSystem)

```
gSystem->Load("libGo4UserAnalysis.so");
// load external library (for CINT)
gSystem->Exec("rm -rf *.root");
// call shell command (from compiled code)
```

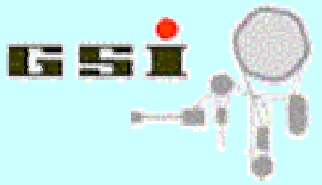
- gDirectory (current root directory, class TDirectory)

```
cout <<gDirectory->GetName()<< endl;
```

- gPad (currently selected draw pad, class TPad)

```
gPad->Clear();
```

Unnamed scripts

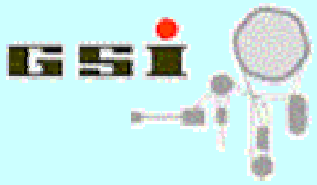


Example hsimple.C (from ROOT tutorials):
http://root.cern.ch/root/html/examples/hsimple.C.html

used ROOT classes:

- **TFile: root file handle object**
- **TCanvas: Window to draw graphics**
- **TH1F, TH2F, TProfile: histogram classes**
- **TNtuple: ntuple of values (simple TTree)**
- **TRandom (global gRandom): random generator**
- **TBenchmark (global gBenchmark):)**

Named scripts



Examples (from Go4 distribution):

http://www-linux.gsi.de/~go4/download/cppworkshop/convertfile.C.html

http://www-linux.gsi.de/~go4/download/cppworkshop/plothistos.C

```
root
root [] .L convertfile.C // load once
root [] convertfile("r113.root"); // call function
root [] convertfile("r114.root"); // call again
root [] .x plothistos("histograms.root",200,700,1);
      // execute with parameters
```

- script must contain one function with same name as scriptfile
- script function may have arguments
- script may contain also other functions / subroutines
- script may contain class definitions
- objects in scope of named scripts are deleted after leaving script,

objects in scope of unnamed scripts remain available!

Automatic Compiler of Libraries for CINT (ACLiC)

```
>root
root [] .x plothistos.C("test_AS",0,2000,1)
```

execute script in interpreter

```
root [] .x plothistos.C+("test_AS",0,2000,1)
```

compile script into library plothistos.C.so in background, then execute function

(note: only recompile if plothistos.C or plothistos.h have changed since last compile)

```
root [] .x plothistos.C++("test_AS",0,2000,1)
```

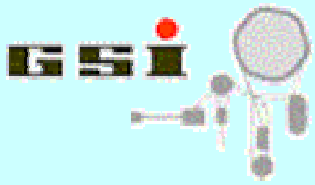
compile library in any case before execution

```
root [] .L convertfile.C+
```

compile script and load library, but do not execute

```
root [] convertfile("testASF");
            // will execute loaded function
```

Compiled code using ROOT



Makefile to build library and executable convertfile:
http://www-linux.gsi.de/~go4/download/cppworkshop/Makefile.html

Main executable:
http://www-linux.gsi.de/~go4/download/cppworkshop/main.cxx.html

required in addition to convertfile.C and convertfile.h

- `make all`
- `...`
- `convertfile test_AS`

C++ differences

CINT interpreted	ACLiC	Compiled code
ROOT class declarations are known implicitely **NOTE: include all non-ROOT classes!**	**ROOT classes require to include declarations, too** **e.g. `#include „TH1.h“`**	**ROOT classes require to include declarations, too** **e.g. `#include „TH1.h“`**
ROOT class definitions are known implicitely	**Automatic linking against ROOT libs and the generated library**	**ROOT classes require to link against respective libraries when building executable** **Makefile**
No function declaration needed	**Declare function before use**	**Declare function before use**
„CINT C++“ (s.a.)	**Compiler C++**	**Compiler C++**

TObject: ROOT top base class

- **defines interface of fundamental virtual methods:**

```
Draw(), Print(), Dump(), GetName(), Clear(),
Compare(), Streamer(), Clone(), Write(),...
```

- **base type for root collections**

```
TObject* ob= new TH1F("hpx","title",2048,0,2047);
TList* list=new TList;
list->Add(ob);
TObject* ob2=list->FindObject("hpx");
```

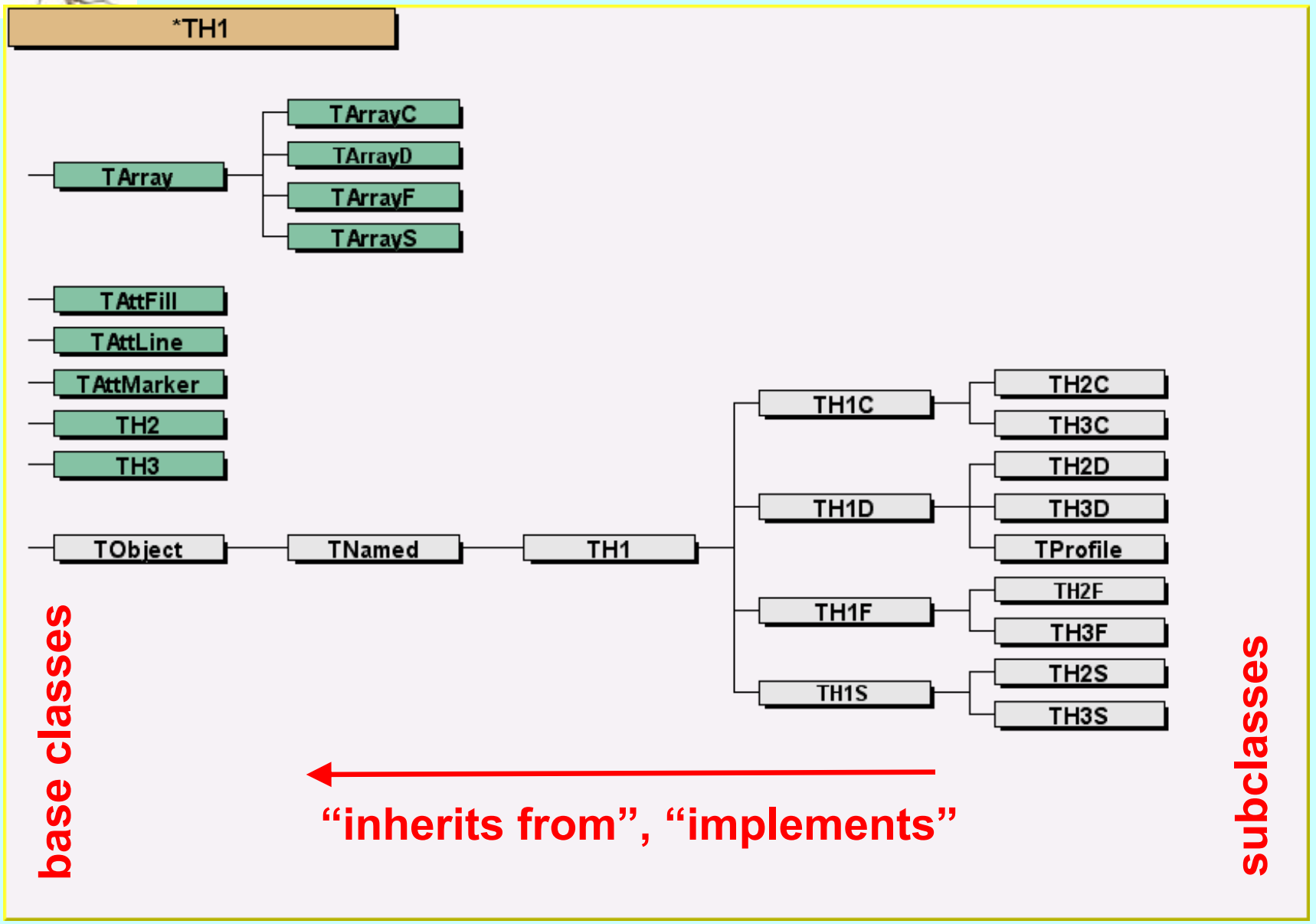
- **IO (via TObject::Streamer()):**

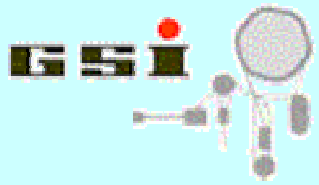
```
ob->Write(); ob->Clone();
```

- **runtime class introspection:**

```
TClass* cl=ob->Class(); // full info on methods
                        and datamembers in memory
if(ob->InheritsFrom("TH1"))... // check type
```

TH1: example of class hierarchy





Class hierarchy: some facts

- **Subclass objects are of parent class type:**

  a TH1D histogram „is a“ TObject

- **Subclasses have all members /methods of parent classes**

```
TH1D* his=new TH1D(“hpx“,“example“,100,0,10);
cout <<“histogram name:“ << his->GetName()<<endl;
```

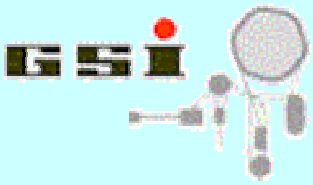
  TH1D uses name property of TNamed

- **Subclasses may redefine virtual methods:**

  TObject::Print() overridden by TH1::Print()

```
TObject* ob=new TH2I(“map“, “example“, 50, 0, 1000, 50, 0, 1000);
ob->Print(); // C++ automatically calls TH2I::Print();
```

Exercises



1. **Understand examples hsimple.C, plothistos.C (see ROOT classes Reference Guide at http://root.cern.ch)**
2. **Write a ROOT script `asciread(const char* filename)` with following tasks:**
   - **read in x,y,z values from an ascii file linewise.**
   - **fill histograms for x, y, x:y and x:y:z from these values and draw them in one TCanvas that is divided into TPads**
   - **fill TTree (TNtuple) with values x:y:z and write this into a TFile**
   - **Example input file: http://www-linux.gsi.de/~go4/download/cppworkshop/input.dat**