ROOT Cheat Sheet

1 Interactive work

Command	Description
`root -h`	show help for `root` command
`root [-l]`	open ROOT
`root [-l] file.root`	open ROOT and load a file
`.?`	show help in interactive ROOT
`.q[qq[qq[qq]]]`	close ROOT (more `q`'s – more violently)
`.ls`	list present ROOT directory
`tree->Show(2)`	show 3rd (!) tree entry
`tree->Scan()`	print entries as a table
`tree->Scan("n:x:y:z", "x>2")`	specify variables to show and condition
`new TBrowser`	open TBrowser
`ntuple->StartViewer()`	open an analysis GUI

2 Scripts/programs

2.1 C++ macro

Running:

```
root [-l] [-b] [-q] macro.C[+[+]]
```

or

```
root [-l] [-b]
.x macro.C[+[+]]
```

```
void macro(){
  cout << "hello, world" << endl;
  TCanvas c("can1", "Canvas title", 800, 600);
  TF1 f("fun1", "sin(x)", -5, 5);
  f.Draw();
  c.SaveAs("sin.png");
}
```

2.2 C++ program

Running:

```
g++ program.C -o program[.exe] `root-config --cflags --glibs`
./program[.exe]
```

```
#include <iostream>
#include "TCanvas.h"
#include "TF1.h"
using namespace std;
int main (){
  cout << "hello, world" << endl;
  TCanvas c("can1", "Canvas title", 800, 600);
  TF1 f("fun1", "sin(x)", -5, 5);
  f.Draw();
  c.SaveAs("sin.png");
  return 0;
}
```

2.3 PyROOT script

```
#!/usr/bin/env python

import ROOT
print "hello, world"
c = ROOT.TCanvas("can1", "Canvas title", 800, 600)
f = ROOT.TF1("fun1", "sin(x)", -5, 5)
f.Draw()
c.SaveAs("sin.png")
```

3 Important classes

3.1 std::vector

Object-oriented array of elements. Not really ROOT, but very useful.

```
#include <vector>
std::vector<double> x;
x.push_back(10); // add element
x.push_back(1.5); // add element
x.push_back(1e2); // add element
std::cout << x.at(1); // get second element
std::cout << x[1]; // alternative way
```

3.2 TFile – reading

```
TFile f("input.root"); // open for reading
TH1F * h = (TH1F*)f.Get("hist1");
TTree * t = (TTree*)f.Get("data");
f.Close;
```

3.3 TFile – writing

```
Tfile f("output.root", "recreate"); // open for writing
TH1F h("h", "title", 100, 0, 10);
h.Fill(10); h.Fill(12); h.Fill(15);
f.Write();
f.Close();
```

3.4 TTree

Class for storing data.

```
TFile f("input.root");
TTree * tree = (TTree*)f.Get("treename");
tree->Draw("x", "y>0"); // Plot x distribution for y > 0
tree->Draw("x:y", "n==1", "colz"); // Plot x vs y
tree->Draw("x>>hist(100, 0, 1)", "y>0"); // named histogram with defined binning
```

3.5 TH1F

One-dimensional histogram.

```
TH1F h("name", "title;x [mm]; number of entries", 100, 0, 1); // constructor
h.Fill(x); // filling the histogram (repeated for each entry)
h.SetLineColor(2);
h.SetLineStyle(3);
h.SetLineWidth(1.5);
h.Scale(10); // Multiply all bins by the same number
h.Integral(); // Calculate sum (!) of bins
h.GetBinContent(10); // Get value of bin 10
h.SetBinContent(10, 0.4); // Set value of bin 10
h.Draw("l"); // drawing
```

3.6 TH2F

Two-dimensional histogram.

```
TH2F h("name", "title;x [mm]; number of entries", 100, 0, 1, 50, -1, 1); // constructor
h.Fill(x, y); // filling the histogram (repeated for each entry)
h.Draw("colz"); // drawing
```

3.7 TGraph

A graph of (x, y) points.

```
double x[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
double y[10] = {0, 1, 4, 9, 16, 25, 36, 49, 64, 81};
TGraph g(10, x, y);
g.Draw("APL"); // Axes, Points, Lines
```

3.8 TCanvas

Canvas on which plots can be drawn.

```
TCanvas c("c1", "Canvas 1", 1024, 768); // constructor
h.Draw(); // drawing a histogram on the current canvas
c.SaveAs("output.png"); // saving to PNG file
c.Print("output.pdf["); // opens a multi-page file
c.Print("output.pdf");  // writes a new page
c.Print("output.pdf]"); // closes the file
c.Divide(2,3); // divide canvas into 6 TPads
c.cd(2); // from now on draw on 2nd sub-canvas
c.SetLogy(); // logarithmic scale on y axis
```

3.9 TLegend

Legend on the plot.

```
TLegend leg(0.7, 0.7, 0.9, 0.9);
leg.AddEntry(h1, "data", "p"); // h1 is a pointer
leg.AddEntry(&h2, "model", "l"); // h2 is an object
```

3.10 TLorentzVector

Useful for calculations in relativistic kinematics.

```
TLorentzVector p1, p2;
p1.SetPxPyPzE(5, 2, 7, 20);
p2.SetPtEtaPhiM(10, 2, 0.5, 2);
double m_inv = (p1 + p2).M();
```

3.11 TF1

One-dimentional function. For plotting and fitting.

```
TF1 f("fun1", "[0] + [1]*x + [2]*sin([3]*x)", -10, 10)
h.Fit(f); // or h.Fit("fun1");
```

3.12 TRandom

```
TRandom gen;
std::cout << gen.Uniform(0, 12);
std::cout << gen.Gaus(2, 0.1);
```