目錄

简介 1.1

常用类的使用 1.2

- FitResult 1.2.1
- TAttAxis* 1.2.2
- TAxis* 1.2.3
- TBackCompFitter 1.2.4
- TBenchmark+ 1.2.5
- TBranch* 1.2.6
- TBuffer+ 1.2.7
- TBufferFile 1.2.8
- TCanvas* 1.2.9
- TChain* 1.2.10
- TCut* 1.2.11
- TCutG* 1.2.12
- TEllipse 1.2.13
- TEventList* 1.2.14
- TF1+ 1.2.15
- TF2 1.2.16
- TF3 1.2.17
- TFile* 1.2.18
- TFileCacheWrite 1.2.19
- TFileMerger 1.2.20
- TFitter 1.2.21
- TFitResult 1.2.22
- TFitResultPtr 1.2.23
- TFractionFitter 1.2.24
- TGaxis* 1.2.25

TGraph*	1.2.26
TGraph2D	1.2.27
TGraph2DErrors	1.2.28
TGraphErrors	1.2.29
TGraphPolar	1.2.30
TH1+	1.2.31
TH2+	1.2.32
TH2Poly	1.2.33
TH3	1.2.34
TLatex*	1.2.35
TLegend*	1.2.36
TLegendEntry*	1.2.37
TLine*	1.2.38
TLinearFitter	1.2.39
TList*	1.2.40
TMath	1.2.41
TMathBase	1.2.42
TMatrixT	1.2.43
TMatrixTBase	1.2.44
TMatrixTSparse	1.2.45
TMatrixTSym	1.2.46
TMemFile	1.2.47
TMinuit	1.2.48
TMinuit2TraceObject	1.2.49
TMinuitMinimizer	1.2.50
TMLPAnalyzer	1.2.51
TMonitor	1.2.52
TMultiGraph+	1.2.53
TMultiLayerPerceptron	1.2.54
TNamed+	1.2.55

TNeuron	1.2.56
TNtuple*	1.2.57
TNtupleD*	1.2.58
TPad*	1.2.59
TPaveStats*	1.2.60
TPolyMarker	1.2.61
TPaveText*	1.2.62
TPolyMarker3D	1.2.63
TProfile	1.2.64
TProfile2D	1.2.65
TProfile3D	1.2.66
TPServerSocket	1.2.67
TPSocket	1.2.68
TRandom+	1.2.69
TRandom1+	1.2.70
TRandom2+	1.2.71
TRandom3+	1.2.72
TROOT+	1.2.73
TServerSocket	1.2.74
TSocket	1.2.75
TSpectrum	1.2.76
TSpectrum2	1.2.77
TSpectrum2Fit	1.2.78
TSpectrum2Painter	1.2.79
TSpectrum2Transform	1.2.80
TSpectrum3	1.2.81
TSpectrumFit	1.2.82
TSpectrumTransform	1.2.83
TSpline+	1.2.84
TStopwatch	1.2.85

TString+ 1.2.86
TStyle+ 1.2.87
TSystem+ 1.2.88
TText* 1.2.89
TThread 1.2.90
TThreadFactory 1.2.91
TThreadImp 1.2.92
TTime* 1.2.93
TTimer* 1.2.94
TTimeStamp 1.2.95
TTree+ 1.2.96
TTreePlayer 1.2.97
TVector2 1.2.98
TVector3 1.2.99
TVectorT 1.2.100
TVirtualFitter 1.2.101
图形界面的使用 1.3
TGButton* 1.3.1
TGButtonGroup* 1.3.2
TGCanvas* 1.3.3
TGClient* 1.3.4
TGColorDialog* 1.3.5
TGColorSelect* 1.3.6
TGComboBox* 1.3.7
TGDoubleSlider* 1.3.8
TGFont* 1.3.9
TGFontDialog 1.3.10
TGFrame* 1.3.11
TGGC 1.3.12
TGIcon* 1.3.13

TGImageMap+	1.3.14
TGLabel*	1.3.15
TGLayout+	1.3.16
TGListBox+	1.3.17
TGListTree+	1.3.18
TGListView+	1.3.19
TGMdiDecorFrame	1.3.20
TGMdiFrame	1.3.21
TGMdiMainFrame	1.3.22
TGMdiMenu	1.3.23
TGMenu+	1.3.24
TGNumberEntry+	1.3.25
TGObject*	1.3.26
TGPicture+	1.3.27
TGProgressBar*	1.3.28
TGScrollBar+	1.3.29
TGSimpleTable+	1.3.30
TGSlider+	1.3.31
TGSpeedo*	1.3.32
TGSplitter*	1.3.33
TGStatusBar*	1.3.34
TGTab+	1.3.35
TGTable+	1.3.36
TGTableCell	1.3.37
TGTableContainer	1.3.38
TGTableHeader	1.3.39
TGTableLayout	1.3.40
TGTextEntry+	1.3.41
TGTextViewStream+	1.3.42
TGTripleSlider+	1.3.43

TGView+ 1.3.44
TGWidget* 1.3.45
TGWindow* 1.3.46
TQObject* 1.3.47
TRootEmbeddedCanvas+ 1.3.48
不常用类的使用 1.4
TDirectory+ 1.4.1
TDirectoryFile+ 1.4.2
TObject+ 1.4.3
TSystemDirectory+ 1.4.4
TSystemFile+ 1.4.5
TTask+ 1.4.6
数据结构 1.5
datarecord 1.5.1
dobject 1.5.2
freesegments 1.5.3
gap 1.5.4
header 1.5.5
keyslist 1.5.6
StreamerInfo 1.5.7
TClonesArray 1.5.8
TDirectory 1.5.9
TFile 1.5.10
TObject 1.5.11
TProcessID 1.5.12
TRefArray 1.5.13
TRef 1.5.14
TTree 1.5.15
预定义 1.6
RVersion 1.6.1

文件版本	1.7

简介

本文为 吴鸿毅（wuhongyi@qq.com） ROOT 使用的一些常用代码及总结。

类的使用

本章主要介绍常用类中的函数

包括：函数的使用、函数的参数设置、示例代码等

```
Char_t      //Signed Character 1 byte
UChar_t     //Unsigned Character 1 byte
Short_t      //Signed Short integer 2 bytes
UShort_t    //Unsigned Short integer 2 bytes
Int_t       //Signed integer 4 bytes
UInt_t      //Unsigned integer 4 bytes
Long64_t    //Portable signed long integer 8 bytes
ULong64_t   //Portable unsigned long integer 8 bytes
Float_t     //Float 4 bytes
Double_t    //Float 8 bytes
Double32_t  //Double 8 bytes in memory, written as a Float 4 byt
es
Bool_t      //Boolean (0=false, 1=true)
```

```
The symbols used for the type are:
C: a character string terminated by the 0 character
B: an 8 bit signed integer
b: an 8 bit unsigned integer
S: a 16 bit signed integer
s: a 16 bit unsigned integer
I: a 32 bit signed integer
i: a 32 bit unsigned integer
L: a 64 bit signed integer
l: a 64 bit unsigned integer
F: a 32 bit floating point
D: a 64 bit floating point
O: [the letter ‘o’, not a zero] a boolean (Bool_t)
```

```
The type of information shown in the histogram statistics box ca
n be selected with:
      gStyle->SetOptStat(mode);

The "mode" has up to nine digits that can be set to on (1 or 2),
 off (0).

      mode = ksiourmen  (default = 000001111)
      k = 1;  kurtosis printed
      k = 2;  kurtosis and kurtosis error printed
      s = 1;  skewness printed
      s = 2;  skewness and skewness error printed
      i = 1;  integral of bins printed
      o = 1;  number of overflows printed
      u = 1;  number of underflows printed
      r = 1;  rms printed
      r = 2;  rms and rms error printed
      m = 1;  mean value printed
      m = 2;  mean and mean error values printed
      e = 1;  number of entries printed
      n = 1;  name of histogram is printed
```

```
gROOT->GetListOfClasses()
gROOT->GetListOfColors()
gROOT->GetListOfTypes()
gROOT->GetListOfGlobals()
gROOT->GetListOfGlobalFunctions()
gROOT->GetListOfFiles()
gROOT->GetListOfMappedFiles()
gROOT->GetListOfSockets()
gROOT->GetListOfCanvases()
gROOT->GetListOfStyles()
gROOT->GetListOfFunctions()
gROOT->GetListOfSpecials()
gROOT->GetListOfGeometries()
gROOT->GetListOfBrowsers()
gROOT->GetListOfMessageHandlers()
//These methods return a TSeqCollection, meaning a collection of
```

```
 objects, and they can be used to do list operations such as fin
ding an object, or traversing the list and calling a method for
each of the members. See the TCollection class description for t
he full set of methods supported for a collection. For example,
to find a canvas called c1you can do:
root[] gROOT->GetListOfCanvases()->FindObject("c1")
//This returns a pointer to a TObject, and before you can use it
 as a canvas you need to cast it to a TCanvas*.


//A graphic object is always drawn on the active pad. It is conv
enient to access the active pad, no matter what it is. For that,
 we have gPad that is always pointing to the active pad. For exa
mple, if you want to change the fill color of the active pad to
blue, but you do not know its name, you can use gPad .
root[] gPad->SetFillColor(38)

// gRandom 默认调用 TRandom3

//gRandom is a pointer to the current random number generator. B
y default, it points to a TRandom3 object, based on the “Mersenn
e-Twister” generator. This generator is very fast and has very g
ood random proprieties (a very long period of 10600 ). Setting t
he seed to 0 implies that the seed will be uniquely generated us
ing the TUUID. Any other value will be used as a constant. The f
ollowing basic random distributions are provided: Rndm() or Unif
orm(min,max), Gaus(mean,sigma), Exp(tau), BreitWigner(mean,sigma
), Landau(mean,sigma), Poisson(mean), Binomial(ntot,prob). You c
an customize your ROOT session by replacing the random number ge
nerator. You can delete gRandom and recreate it with your own. F
or example:
root[] delete gRandom;
root[] gRandom = new TRandom2(0); //seed=0
//TRandom2 is another generator, which is also very fast and use
s only three words for its state.


//gFile is the pointer to the current opened file in the ROOT se
ssion.
```

```
//gDirectory is a pointer to the current directory. The concept
and role of a directory is explained in the chapter “Input/Outpu
t”.
```

画图参数

```
//Fitting Histograms

• *fname:The name of the fitted function (the model) is passed a
s the first parameter. This name may be one of ROOT pre-defined
function names or a user-defined function. The functions below a
re predefined, and can be used with the TH1::Fit method:
• “gaus” Gaussian function with 3 parameters: f(x) = p0*exp(-0.5
*((x-p1)/p2)^2)
• “expo”An Exponential with 2 parameters: f(x) = exp(p0+p1*x)
• “polN ” A polynomial of degree N : f(x) = p0 + p1*x + p2*x2 +.
..
• “landau” Landau function with mean and sigma. This function ha
s been adaptedfrom the CERNLIB routine G110 denlan.
• *option:The second parameter is the fitting option. Here is th
e list of fitting options:
• “W” Set all weights to 1 for non empty bins; ignore error bars
• “WW” Set all weights to 1 including empty bins; ignore error b
ars
• “I” Use integral of function in bin instead of value at bin ce
nter
• “L” Use log likelihood method (default is chi-square method)
• “U” Use a user specified fitting algorithm
• “Q” Quiet mode (minimum printing)
• “V” Verbose mode (default is between Q and V)
• “E” Perform better errors estimation using the Minos technique
• “M” Improve fit results
• “R” Use the range specified in the function range
• “N” Do not store the graphics function, do not draw
• “0” Do not plot the result of the fit. By default the fitted f
unction is drawn unless the option “N” above is specified.
• “+” Add this new fitted function to the list of fitted functio
ns (by default, the previous function is deleted and only the la
```

```
st one is kept)
• “B”Use this option when you want to fix one or more parameters
 and the fitting function is like polN, expo, landau, gaus.
• “LL”An improved Log Likelihood fit in case of very low statist
ics and when bincontentsare not integers. Do not use this option
 if bin contents are large (greater than 100).
• “C”In case of linear fitting, don’t calculate the chisquare (s
aves time).
• “F”If fitting a polN, switch to Minuit fitter (by default, pol
N functions are fitted by the linear fitter).
• *goption:The third parameter is the graphics option that is th
e same as in the TH1::Draw (see the chapter Draw Options).
• xxmin, xxmax:Thee fourth and fifth parameters specify the rang
e over which to apply the fit.
```

```
//Graph Draw Options
The various drawing options for a graph are explained in TGraph:
:PaintGraph. They are:
• “L” A simple poly-line between every points is drawn   //折线图
,将每个bin用线连接
• “F” A fill area is drawn
• “F1” Idem as “F” but fill area is no more repartee around X=0
or Y=0
• “F2” draw a fill area poly line connecting the center of bins
• “A” Axis are drawn around the graph
• “C” A smooth curve is drawn  //将每个bin用一条光滑的曲线连接起来
• “*” A star is plotted at each point//每个点用*表示
• “P” The current marker of the graph is plotted at each point
• “B” A bar chart is drawn at each point
• “[]” Only the end vertical/horizontal lines of the error bars
are drawn. This option only applies to the TGraphAsymmErrors.
• “1” ylow = rwymin

The TGraphPolar drawing options are:
“O” Polar labels are paint orthogonally to the polargram radius.
“P” Polymarker are paint at each point position.
“E” Paint error bars.
“F” Paint fill area (closed polygon).
“A”Force axis redrawing even if a polagram already exists.
```

```
//Draw Options                    //eg. h->Draw("E1SAME"); 或 h->
SetOption("lego");h->Draw();
The following draw options are supported on all histogram classe
s:
• “AXIS”: Draw only the axis.
• “HIST”: When a histogram has errors, it is visualized by defau
lt with error bars. To visualize it without errors use HIST toge
ther with the required option (e.g. “HIST SAME C”).
• “SAME”: Superimpose on previous picture in the same pad.
• “CYL”: Use cylindrical coordinates.
• “POL”: Use polar coordinates.
• “SPH”: Use spherical coordinates.
• “PSR”: Use pseudo-rapidity/phi coordinates.  //通常的二维直方图
```

```
• “LEGO”: Draw a lego plot with hidden line removal.
• “LEGO1”: Draw a lego plot with hidden surface removal.
• “LEGO2”: Draw a lego plot using colors to show the cell conten
ts.
• “SURF”: Draw a surface plot with hidden line removal.
• “SURF1”: Draw a surface plot with hidden surface removal.
• “SURF2”: Draw a surface plot using colors to show the cell con
tents.
• “SURF3”: Same as SURF with a contour view on the top.
• “SURF4”: Draw a surface plot using Gouraud shading.
• “SURF5”: Same as SURF3 but only the colored contour is drawn.
Used with option CYL , SPH or PSR it allows to draw colored cont
ours on a sphere, a cylinder or in a pseudo rapidly space. In Ca
rtesian or polar coordinates,option SURF3 is used.

The following options are supported for 1-D histogram classes:
• “AH”: Draw the histogram, but not the axis labels and tick mar
ks
• “B”: Draw a bar chart
• “C”: Draw a smooth curve through the histogram bins  //将每个bi
n用一条光滑的曲线连接起来
• “E”: Draw the error bars    //“E0”- “E4” 与误差有关的参数
• “E0”: Draw the error bars including bins with 0 contents
• “E1”: Draw the error bars with perpendicular lines at the edge
s  //垂直方向的误差棒
• “E2”: Draw the error bars with rectangles
• “E3”: Draw a fill area through the end points of the vertical
error bars
• “E4”: Draw a smoothed filled area through the end points of th
e error bars
• “L”: Draw a line through the bin contents  //将每个bin用线连接
• “P”: Draw a (poly)marker at each bin using the histogram’s cur
rent marker style
• “P0”: Draw current marker at each bin including empty bins
• “PIE”: Draw a Pie Chart
• “*H”: Draw histogram with a * at each bin
• “LF2”: Draw histogram as with option “L” but with a fill area.
 Note that “L” also draws a fill area if the histogram fill colo
r is set but the fill area corresponds to the histogram contour.
• “9”: Force histogram to be drawn in high resolution mode. By d
```

```
efault, the histogram is drawn in low resolution in case the num
ber of bins is greater than the number of pixels in the current
pad
• “][”: Draw histogram without the vertical lines for the first
and the last bin. Use it when superposing many histograms on the
 same picture.

The following options are supported for 2-D histogram classes:
• “ARR”: Arrow mode. Shows gradient between adjacent cells
• “BOX”: Draw a box for each cell with surface proportional to c
ontents  //每个单元画一BOX，Box面积正比于bin contents
• “BOX1”: A sunken button is drawn for negative values, a raised
 one for positive values
• “COL”: Draw a box for each cell with a color scale varying wit
h contents  //每个单元画一个box，颜色与 bin contents相关
• “COLZ”: Same as “COL” with a drawn color palette  //同col，但是
画一个条显示颜色与内容对应关系
• “CONT”: Draw a contour plot (same as CONT0 )  //画轮廓图
• “CONTZ”: Same as “CONT” with a drawn color palette
• “CONT0”: Draw a contour plot using surface colors to distingui
sh contours
• “CONT1”: Draw a contour plot using line styles to distinguish
contours
• “CONT2”: Draw a contour plot using the same line style for all
 contours
• “CONT3”: Draw a contour plot using fill area colors
• “CONT4”: Draw a contour plot using surface colors (SURF2 optio
n at theta = 0)
• "CONT5": Use Delaunay triangles to compute the contours
• “LIST”: Generate a list of TGraph objects for each contour
• “FB”: To be used with LEGO or SURFACE , suppress the Front-Box
• “BB”: To be used with LEGO or SURFACE , suppress the Back-Box
• “A”: To be used with LEGO or SURFACE , suppress the axis
• “SCAT”: Draw a scatter-plot (default)  //绘制散点图
• “SPEC”: Use TSpectrum2Painter tool for drawing
• “TEXT”: Draw bin contents as text (format set via gStyle->SetP
aintTextFormat) .
• “TEXTnn”: Draw bin contents as text at angle nn ( 0<nn<90 ).
• “[cutg]”: Draw only the sub-range selected by the TCutG name “
cutg”.
```

```
• “Z”: The “Z” option can be specified with the options: BOX, CO
L, CONT, SURF, and LEGO to display the color palette with an axi
s indicating the value of the corresponding color on the right s
ide ofthe picture.

The following options are supported for 3-D histogram classes:
• " " : Draw a 3D scatter plot.
• “BOX”: Draw a box for each cell with volume proportional to co
ntents
• “LEGO”: Same as “BOX”
• “ISO”: Draw an iso surface
• “FB”: Suppress the Front-Box
• “BB”: Suppress the Back-Box
• “A”: Suppress the axis
```

未分类

```
//支持中文显示
"\\hbox{RHIC スピン物理 Нью-Йорк}"
```

```
//直方图上不要TPave
gStyle->SetOptStat(0);
```

```
char name[10], title[20];
TObjArray Hlist(0);
TH1F* h;
for (Int_t i = 0; i < 15; i++) {
sprintf(name,"h%d",i);   //把格式化的数据写入字符串中
sprintf(title,"histo nr:%d",i);
h = new TH1F(name,title,100,-4,4);
Hlist.Add(h);
h->FillRandom("gaus",1000);}
TFile f("demo.root","recreate");
Hlist->Write();
f.Close();
```

```
TH3D *h3 = new TH3D("h3", "h3", 20, -2, 2, 20, -2, 2, 20, 0, 4);
Double_t x,y,z;
for (Int_t i=0; i<10000; i++) {
gRandom->Rannor(x,y);
z=x*x+y*y;
h3->Fill(x,y,z);
}
h3->Draw("iso");

TF3 *fun3 = new TF3("fun3","sin(x*x+y*y+z*z-36)",-2,2,-2,2,-2,2)
;
fun3->Draw();
```

```
//进度滑动条
TCanvas *c1 = new TCanvas("c1","The HSUM example",200,10,600,400
);
c1->SetGrid();
gBenchmark->Start("hsum");
// Create some histograms.
auto total  = new TH1F("total","This is the total distribution",
100,-4,4);
auto main   = new TH1F("main","Main contributor",100,-4,4);
auto s1     = new TH1F("s1","This is the first signal",100,-4,4)
;
auto s2     = new TH1F("s2","This is the second signal",100,-4,4
);
total->Sumw2();  // store the sum of squares of weights
total->SetMarkerStyle(21);
total->SetMarkerSize(0.7);
main->SetFillColor(16);
s1->SetFillColor(42);
s2->SetFillColor(46);
TSlider *slider = 0;////////滑动条

// Fill histograms randomly
gRandom->SetSeed();
const Int_t kUPDATE = 500;
Float_t xs1, xs2, xmain;
```

```
for ( Int_t i=0; i<10000; i++) {
  xmain = gRandom->Gaus(-1,1.5);
  xs1   = gRandom->Gaus(-0.5,0.5);
  xs2   = gRandom->Landau(1,0.15);
  main->Fill(xmain);
  s1->Fill(xs1,0.3);
  s2->Fill(xs2,0.2);
  total->Fill(xmain);
  total->Fill(xs1,0.3);
  total->Fill(xs2,0.2);
  if (i && (i%kUPDATE) == 0) {
    if (i == kUPDATE) {
      total->Draw("e1p");
      main->Draw("same");
      s1->Draw("same");
      s2->Draw("same");
      c1->Update();
      slider = new TSlider("slider",
               "test",4.2,0,4.6,total->GetMaximum(),38);
      slider->SetFillColor(46);
    }
    if (slider) slider->SetRange(0,Float_t(i)/10000.);
    c1->Modified();
    c1->Update();
  }
 }
slider->SetRange(0,1);
total->Draw("sameaxis"); // to redraw axis hidden by the fill ar
ea
c1->Modified();
gBenchmark->Show("hsum");
```

```
TPaveLabel pl;

//basic 2-d options
Float_t xMin=0.67, yMin=0.875, xMax=0.85, yMax=0.95;
Int_t cancolor = 17;
TCanvas c2h("c2h","2-d options",10,10,800,600);
c2h.Divide(2,2);
```

```
c2h.SetFillColor(cancolor);
c2h.cd(1);
h2.Draw();       pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"SCAT","br
NDC");
c2h.cd(2);
h2.Draw("box");  pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"BOX","brN
DC");
c2h.cd(3);
h2.Draw("arr");  pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"ARR","brN
DC");
c2h.cd(4);
h2.Draw("colz"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"COLZ","br
NDC");
c2h.Update();

//text option
TCanvas ctext("ctext","text option",50,50,800,600);
gPad->SetGrid();
ctext.SetFillColor(cancolor);
ctext.SetGrid();
h2.Draw("text"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"TEXT","br
NDC");
ctext.Update();

//contour options
TCanvas cont("contours","contours",100,100,800,600);
cont.Divide(2,2);
gPad->SetGrid();
cont.SetFillColor(cancolor);
cont.cd(1);
h2.Draw("contz"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"CONTZ","
brNDC");
cont.cd(2);
gPad->SetGrid();
h2.Draw("cont1"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"CONT1","
brNDC");
cont.cd(3);
gPad->SetGrid();
h2.Draw("cont2"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"CONT2","
brNDC");
```

```
cont.cd(4);
gPad->SetGrid();
h2.Draw("cont3"); pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"CONT3","
brNDC");
cont.Update();

//lego options
TCanvas lego("lego","lego options",150,150,800,600);
lego.Divide(2,2);
lego.SetFillColor(cancolor);
lego.cd(1);
h2.Draw("lego");     pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"LEGO",
"brNDC");
lego.cd(2);
h2.Draw("lego1");    pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"LEGO1"
,"brNDC");
lego.cd(3);
gPad->SetTheta(61); gPad->SetPhi(-82);
h2.Draw("surf1pol"); pl.DrawPaveLabel(xMin,yMin,xMax+0.05,yMax,"
SURF1POL","brNDC");
lego.cd(4);
gPad->SetTheta(21); gPad->SetPhi(-90);
h2.Draw("surf1cyl"); pl.DrawPaveLabel(xMin,yMin,xMax+0.05,yMax,"
SURF1CYL","brNDC");
lego.Update();

//surface options
TCanvas surf("surfopt","surface options",200,200,800,600);
surf.Divide(2,2);
surf.SetFillColor(cancolor);
surf.cd(1);
h2.Draw("surf1");   pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"SURF1",
"brNDC");
surf.cd(2);
h2.Draw("surf2z");  pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"SURF2Z"
,"brNDC");
surf.cd(3);
h2.Draw("surf3");   pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"SURF3",
"brNDC");
surf.cd(4);
```

```
h2.Draw("surf4");   pl.DrawPaveLabel(xMin,yMin,xMax,yMax,"SURF4",
"brNDC");
surf.Update();
```

FitResult

TAttAxis

基类

class

```
   TAttAxis();
   virtual          ~TAttAxis();
   void     Copy(TAttAxis &attaxis) const;/// Copy of the object.

   virtual Int_t    GetNdivisions()  const {return fNdivisions;}
   virtual Color_t  GetAxisColor()   const {return fAxisColor;}
   virtual Color_t  GetLabelColor()  const {return fLabelColor;}
   virtual Style_t  GetLabelFont()   const {return fLabelFont;}
   virtual Float_t  GetLabelOffset() const {return fLabelOffset;
}
   virtual Float_t  GetLabelSize()   const {return fLabelSize;}
   virtual Float_t  GetTitleOffset() const {return fTitleOffset;
}
   virtual Float_t  GetTitleSize()   const {return fTitleSize;}
   virtual Float_t  GetTickLength()  const {return fTickLength;}
   virtual Color_t  GetTitleColor()  const {return fTitleColor;}
   virtual Style_t  GetTitleFont()   const {return fTitleFont;}
   virtual void     ResetAttAxis(Option_t *option="");/// Reset
axis attributes
   virtual void     SaveAttributes(std::ostream &out, const char
 *name, const char *subname);/// Save axis attributes as C++ sta
tement(s) on output stream out

/// Set the number of divisions for this axis.
///
///  - if optim = kTRUE (default), the number of divisions will
be
///                      optimized around the specified value.
///  - if optim = kFALSE, or n < 0, the axis will be forced to u
se
///                      exactly n divisions.
```

```
/// n = n1 + 100*n2 + 10000*n3
/// Where n1 is the number of primary divisions,
/// n2 is the number of second order divisions and
/// n3 is the number of third order divisions.
/// If the number of divisions is "optimized" (see above) n1, n2
, n3 are maximum values.
   virtual void     SetNdivisions(Int_t n=510, Bool_t optim=kTRU
E);   // *MENU*
   virtual void     SetNdivisions(Int_t n1, Int_t n2, Int_t n3,
Bool_t optim=kTRUE);
   virtual void     SetAxisColor(Color_t color=1, Float_t alpha=
1.);  // *MENU* /// Set color of the line axis and tick marks
   virtual void     SetLabelColor(Color_t color=1, Float_t alpha=
1.); // *MENU* /// Set color of labels
   virtual void     SetLabelFont(Style_t font=62);
      // *MENU* /// Set labels' font.

/// Set distance between the axis and the labels
/// The distance is expressed in per cent of the pad width
   virtual void     SetLabelOffset(Float_t offset=0.005);
      // *MENU*
   virtual void     SetLabelSize(Float_t size=0.04);
      // *MENU* /// Set size of axis labels. The size is express
ed in per cent of the pad width
   virtual void     SetTickLength(Float_t length=0.03);
      // *MENU* /// Set tick mark length. The length is expresse
d in per cent of the pad width
   virtual void     SetTickSize(Float_t size=0.03) {SetTickLengt
h(size);}

/// Set distance between the axis and the axis title
/// Offset is a correction factor with respect to the "standard"
 value.
///  - offset = 1   uses the default position that is computed i
n function
///                 of the label offset and size.
///  - offset = 1.2 will add 20 per cent more to the default off
set.
   virtual void     SetTitleOffset(Float_t offset=1);
      // *MENU*
```

```
   virtual void     SetTitleSize(Float_t size=0.04);
      // *MENU* /// Set size of axis title. The size is expresse
d in per cent of the pad width
   virtual void     SetTitleColor(Color_t color=1);
      // *MENU* /// Set color of axis title
   virtual void     SetTitleFont(Style_t font=62);
      // *MENU* /// Set the title font.
```

code

example

TAxis

继承 TNamed, TAttAxis

This class manages histogram axis. It is referenced by TH1 and TGraph. To make a graphical representation of an histogram axis, this class references the TGaxis class. TAxis supports axis with fixed or variable bin sizes. Labels may be associated to individual bins.

class

```
// TAxis status bits
enum {
       kDecimals      = BIT(7),
       kTickPlus      = BIT(9),
       kTickMinus     = BIT(10),
       kAxisRange     = BIT(11),
       kCenterTitle   = BIT(12),
       kCenterLabels  = BIT(14), //bit 13 is used by TObject
       kRotateTitle   = BIT(15),
       kPalette       = BIT(16),
       kNoExponent    = BIT(17),
       kLabelsHori    = BIT(18),
       kLabelsVert    = BIT(19),
       kLabelsDown    = BIT(20),
       kLabelsUp      = BIT(21),
       kIsInteger     = BIT(22),
       kMoreLogLabels = BIT(23)
};

TAxis();
TAxis(Int_t nbins, Double_t xmin, Double_t xmax);
TAxis(Int_t nbins, const Double_t *xbins);
TAxis(const TAxis &axis);
virtual ~TAxis();
TAxis& operator=(const TAxis&);
```

```
   Bool_t      CanExtend() const { return (fBits2 & kCanExtend);
 }
   void        SetCanExtend(Bool_t canExtend) { fBits2 = canExten
d ? (fBits2 | kCanExtend) : (fBits2 & ~kCanExtend); }
   void        SetNoAlphanumeric(Bool_t noalpha = kTRUE) {
      fBits2 = noalpha ? (fBits2 | kNotAlpha) : (fBits2 & ~kNotA
lpha);
      if (IsAlphanumeric() ) {
         SetCanExtend(kFALSE);
         SetAlphanumeric(kFALSE);
      }
   }

/// Center axis labels. If center = kTRUE axis labels will be ce
ntered
/// (hori axes only) on the bin center default is to center on t
he primary tick marks
/// This option does not make sense if there are more bins than
tick marks
   void                CenterLabels(Bool_t center=kTRUE);

/// Center axis title. If center = kTRUE axis title will be cent
ered
/// default is right adjusted
   void                CenterTitle(Bool_t center=kTRUE);

/// Choose a reasonable time format from the coordinates in the
active pad and the number of divisions in this axis
/// If orientation = "X", the horizontal axis of the pad will be
 used for ref.
/// If orientation = "Y", the vertical axis of the pad will be u
sed for ref.
   const char         *ChooseTimeFormat(Double_t axislength=0);
   virtual void        Copy(TObject &axis) const;/// Copy axis st
ructure to another axis
   virtual void        Delete(Option_t * /*option*/ ="") { }
   virtual Int_t       DistancetoPrimitive(Int_t px, Int_t py);//
/ Compute distance from point px,py to an axis
   virtual TObject    *DrawClone(Option_t * /*option*/ ="") const
{return 0;}
```

```
/// Execute action corresponding to one event
///  This member function is called when an axis is clicked with
 the locator.
///  The axis range is set between the position where the mouse
is pressed and the position where it is released.
///  If the mouse position is outside the current axis range whe
n it is released the axis is unzoomed with the corresponding pro
portions.
///  Note that the mouse does not need to be in the pad or even
canvas when it is released.
   virtual void       ExecuteEvent(Int_t event, Int_t px, Int_t
py);

/// Find bin number corresponding to abscissa x. NOTE: this meth
od does not work with alphanumeric bins !!!
/// If x is underflow or overflow, attempt to extend the axis if
 TAxis::kCanExtend is true. Otherwise, return 0 or fNbins+1.
   virtual Int_t      FindBin(Double_t x);
   virtual Int_t      FindBin(Double_t x) const { return FindFix
Bin(x); }

/// Find bin number with label.
/// If the List of labels does not exist create it and make the
axis alphanumeric
/// If one wants just to add a single label- just call TAxis::Se
tBinLabel
/// If label is not in the list of labels do the following depen
ding on the
/// bit TAxis::kCanExtend; of the axis.
///   - if the bit is set add the new label and if the number of
 labels exceeds
///      the number of bins, double the number of bins via TH1::
LabelsInflate
///   - if the bit is not set and the histogram has labels in ea
ch bin
///        set the bit automatically and consider the histogram
as alphanumeric
///     if histogram has only some bins with labels then the his
togram is not
```

```
///     consider alphanumeric and return -1
/// -1 is returned only when the Axis has no parent histogram
   virtual Int_t      FindBin(const char *label);

/// Find bin number corresponding to abscissa x
/// Identical to TAxis::FindBin except that if x is an underflow
/overflow no attempt is made to extend the axis.
   virtual Int_t      FindFixBin(Double_t x) const;

/// Find bin number with label.
/// If the List of labels does not exist or the label doe not ex
ist just return -1 .
/// Do not attempt to modify the axis. This is different than Fi
ndBin
   virtual Int_t      FindFixBin(const char *label) const;
   virtual Double_t   GetBinCenter(Int_t bin) const;/// Return c
enter of bin

/// Return center of bin in log
/// With a log-equidistant binning for a bin with low and up edg
es, the mean is :
/// 0.5*(ln low + ln up) i.e. sqrt(low*up) in logx (e.g. sqrt(10
^0*10^2) = 10).
/// Imagine a bin with low=1 and up=100 :
/// - the center in lin is (100-1)/2=50.5
/// - the center in log would be sqrt(1*100)=10 (!=log(50.5))
/// NB: if the low edge of the bin is negative, the function ret
urns the bin center
///     as computed by TAxis::GetBinCenter
   virtual Double_t   GetBinCenterLog(Int_t bin) const;
   const char        *GetBinLabel(Int_t bin) const;/// Return la
bel for bin
   virtual Double_t   GetBinLowEdge(Int_t bin) const;/// Return
low edge of bin
   virtual Double_t   GetBinUpEdge(Int_t bin) const;/// Return u
p edge of bin
   virtual Double_t   GetBinWidth(Int_t bin) const;/// Return bi
n width
   virtual void       GetCenter(Double_t *center) const;/// Retu
rn an array with the center of all bins
```

```
            Bool_t      GetCenterLabels() const { return TestBit(k
CenterLabels); }
            Bool_t      GetCenterTitle() const { return TestBit(kC
enterTitle); }
            Bool_t      GetDecimals() const { return TestBit(kDeci
mals); }
   THashList         *GetLabels() const { return fLabels; }
   virtual void       GetLowEdge(Double_t *edge) const;/// Retur
n an array with the lod edge of all bins
            Bool_t      GetMoreLogLabels() const { return TestBit(
kMoreLogLabels); }
            Int_t       GetNbins() const { return fNbins; }
            Bool_t      GetNoExponent() const { return TestBit(kNo
Exponent); }
   virtual TObject   *GetParent() const {return fParent;}
            Bool_t      GetRotateTitle() const { return TestBit(kR
otateTitle); }
   virtual const char *GetTicks() const;/// Return the ticks opt
ion (see SetTicks)
   virtual Bool_t      GetTimeDisplay() const {return fTimeDispla
y;}
   virtual const char *GetTimeFormat() const {return fTimeFormat
.Data();}
   virtual const char *GetTimeFormatOnly() const;/// Return *onl
y* the time format from the string fTimeFormat
   const char        *GetTitle() const {return fTitle.Data();}
   const TArrayD     *GetXbins() const {return &fXbins;}

/// Return first bin on the axis
/// i.e. 1 if no range defined
/// NOTE: in some cases a zero is returned (see TAxis::SetRange)
            Int_t       GetFirst() const;

/// Return last bin on the axis
/// i.e. fNbins if no range defined
/// NOTE: in some cases a zero is returned (see TAxis::SetRange)
            Int_t       GetLast() const;
            Double_t    GetXmin() const {return fXmin;}
            Double_t    GetXmax() const {return fXmax;}
   virtual void       ImportAttributes(const TAxis *axis);/// Co
```

```
py axis attributes to this
   Bool_t              IsVariableBinSize() const {
                          // true if axis has variable bin sizes,
 false otherwise
                          return (fXbins.GetSize() != 0);
                       }

///  Set option(s) to draw axis with labels
///  option = "a" sort by alphabetic order
///         = ">" sort by decreasing values
///         = "<" sort by increasing values
///         = "h" draw labels horizonthal
///         = "v" draw labels vertical
///         = "u" draw labels up (end of label right adjusted)
///         = "d" draw labels down (start of label left adjusted)

   virtual void       LabelsOption(Option_t *option="h");  // *M
ENU*

/// Rotate title by 180 degrees. By default the title is drawn r
ight adjusted.
/// If rotate is TRUE, the title is left adjusted at the end of
the axis and rotated by 180 degrees
           void       RotateTitle(Bool_t rotate=kTRUE); // *TOGG
LE* *GETTER=GetRotateTitle
   virtual void       SaveAttributes(std::ostream &out, const ch
ar *name, const char *subname);/// Save axis attributes as C++ s
tatement(s) on output stream out
   virtual void       Set(Int_t nbins, Double_t xmin, Double_t x
max);/// Initialize axis with fix bins
   virtual void       Set(Int_t nbins, const Float_t *xbins);///
 Initialize axis with variable bins
   virtual void       Set(Int_t nbins, const Double_t *xbins);//
/ Initialize axis with variable bins

/// Set label for bin.
/// If no label list exists, it is created. If all the bins have
 labels, the
/// axis becomes alphanumeric and extendable.
/// New labels will not be added with the Fill method but will e
```

```
nd-up in the
/// underflow bin. See documentation of TAxis::FindBin(const cha
r*)
   virtual void       SetBinLabel(Int_t bin, const char *label);
//设置bin名称

/// Sets the decimals flag
/// By default, blank characters are stripped, and then the labe
l is correctly aligned.
/// If the dot is the last character of the string, it is also s
tripped, unless this option is specified.
           void       SetDecimals(Bool_t dot = kTRUE); // *TOGGL
E* *GETTER=GetDecimals
   virtual void       SetDefaults();/// Set axis default values
(from TStyle)
   virtual void       SetDrawOption(Option_t * /*option*/ ="") {
 }
   virtual void       SetLimits(Double_t xmin, Double_t xmax) {
/* set axis limits */ fXmin = xmin; fXmax = xmax; }

/// Set the kMoreLogLabels bit flag
/// When this option is selected more labels are drawn when in l
og scale and there is a small number
/// of decades  (<3).
/// The flag (in fBits) is passed to the drawing function TGaxis
::PaintAxis
   void       SetMoreLogLabels(Bool_t more=kTRUE);  // *TOGGLE*
*GETTER=GetMoreLogLabels

/// Set the NoExponent flag
/// By default, an exponent of the form 10^N is used when the la
bel value are either all very small or very large.
/// The flag (in fBits) is passed to the drawing function TGaxis
::PaintAxis
           void       SetNoExponent(Bool_t noExponent=kTRUE);  /
/ *TOGGLE* *GETTER=GetNoExponent
   virtual void       SetParent(TObject *obj) {fParent = obj;}

///  Set the viewing range for the axis from bin first to last.
///  To set a range using the axis coordinates, use TAxis::SetRa
```

```
ngeUser.
   virtual void       SetRange(Int_t first=0, Int_t last=0);  //
 *MENU*

///  Set the viewing range for the axis from ufirst to ulast (in
 user coordinates).
///  To set a range using the axis bin numbers, use TAxis::SetRa
nge.
   virtual void       SetRangeUser(Double_t ufirst, Double_t ula
st);  // *MENU* 人为设置坐标范围！

///  Set ticks orientation.
///  option = "+"  ticks drawn on the "positive side" (default)
///  option = "-"  ticks drawn on the "negative side"
///  option = "+-" ticks drawn on both sides
   virtual void       SetTicks(Option_t *option="+"); // *MENU*
 //  option = "+"  ticks drawn on the "positive side" (default)
; option = "-"  ticks drawn on the "negative side" ; option = "+
-" ticks drawn on both sides
   virtual void       SetTimeDisplay(Int_t value) {fTimeDisplay
= (value != 0);} // *TOGGLE*

/// Change the format used for time plotting
/// The format string for date and time use the same options as
the one used
/// in the standard strftime C function, i.e. :
/// for date :
///          %a abbreviated weekday name
///          %b abbreviated month name
///          %d day of the month (01-31)
///          %m month (01-12)
///          %y year without century
/// for time :
///
///          %H hour (24-hour clock)
///          %I hour (12-hour clock)
///          %p local equivalent of AM or PM
///          %M minute (00-59)
///          %S seconds (00-61)
///          %% %
```

```
/// This function allows also to define the time offset. It is d
one via %F
/// which should be appended at the end of the format string. Th
e time
/// offset has the following format: 'yyyy-mm-dd hh:mm:ss'
   virtual void       SetTimeFormat(const char *format="");  //
*MENU*

/// Change the time offset
/// If option = "gmt", set display mode to GMT.
   virtual void       SetTimeOffset(Double_t toffset, Option_t *
option="local");
   virtual void       UnZoom();  // *MENU* /// Reset first & las
t bin to the full range

/// Zoom out by a factor of 'factor' (default =2)
///   uses previous zoom factor by default
/// Keep center defined by 'offset' fixed
///   ie. -1 at left of current range, 0 in center, +1 at right
   virtual void       ZoomOut(Double_t factor=0, Double_t offset=
0);  // *MENU*
```

code

```
//设置坐标范围
TAxis *axis=h1->GetXaxis();
axis->SetRangeUser(950,1300);
TAxis *axisy=h1->GetYaxis();
axisy->SetRangeUser(0,100);


//直方图坐标设置 TAttAxis.h
h3->SetTitle("");
h3->GetXaxis()->SetNdivisions(611);//坐标刻度设置，n = n1 + 100*n2
 + 10000*n3 ，Where n1 is the number of primary divisions, n2 is
 the number of second order divisions and n3 is the number of th
ird order divisions.
h3->GetXaxis()->SetTitleSize(20);
h3->GetXaxis()->SetTitleFont(43);
h3->GetXaxis()->SetTitleOffset(3.);//调节title与坐标轴的距离
h3->GetXaxis()->SetLabelFont(44);
h3->GetXaxis()->SetLabelSize(20);
h3->GetYaxis()->SetTitle("ratio h1/h2 ");
h3->GetYaxis()->SetNdivisions(505);
h3->GetYaxis()->SetTitleSize(20);
h3->GetYaxis()->SetTitleFont(43);
h3->GetYaxis()->SetTitleOffset(1.55);//调节title与坐标轴的距离
h3->GetYaxis()->SetLabelFont(43);
h3->GetYaxis()->SetLabelSize(15);


h1->SetLineColor(1);
//1 black;2 red;3 green;4 blue;5yellow;6 magenta;……
h1->SetLogx();   //设置x轴为对数坐标轴
h1->GetXaxis()->SetTitle("x");
h->GetXaxis()->SetTitle("X axis title");
h->GetYaxis()->SetTitle("Y axis title");
h->GetZaxis()->SetTitle("Z axis title");
h1->SetTitle("title name;X axis;Y axis");
h->GetXaxis()->CenterTitle();
h->GetYaxis()->CenterTitle();

h1->SetFrameFillColor(33);    //设置图片背景颜色
```

```
/// Change the format used for time plotting
/// The format string for date and time use the same options as
the one used
/// in the standard strftime C function, i.e. :
/// for date :
///          %a abbreviated weekday name
///          %b abbreviated month name
///          %d day of the month (01-31)
///          %m month (01-12)
///          %y year without century
/// for time :
///          %H hour (24-hour clock)
///          %I hour (12-hour clock)
///          %p local equivalent of AM or PM
///          %M minute (00-59)
///          %S seconds (00-61)
///          %% %
/// This function allows also to define the time offset. It is d
one via %F
/// which should be appended at the end of the format string. Th
e time
/// offset has the following format: 'yyyy-mm-dd hh:mm:ss'
/// Example:
h = new TH1F("Test","h",3000,0.,200000.);
h->GetXaxis()->SetTimeDisplay(1);
h->GetXaxis()->SetTimeFormat("%d\/%m\/%y%F2000-02-28 13:00:01");

/// This defines the time format being "dd/mm/yy" and the time o
ffset as the
/// February 28th 2003 at 13:00:01
/// If %F is not specified, the time offset used will be the one
 defined by:
/// gStyle->SetTimeOffset. For example like that:

TDatime da(2003,02,28,12,00,00);
gStyle->SetTimeOffset(da.Convert());
```

example

TBackCompFitter

TBenchmark

系统自带计时

继承 TNamed

class

```
Int_t                GetBench(const char *name) const;
Float_t              GetCpuTime(const char *name);
Float_t              GetRealTime(const char *name);
virtual void         Print(Option_t *name="") const;
virtual void         Reset();
virtual void         Show(const char *name);
virtual void         Start(const char *name);
virtual void         Stop(const char *name);
virtual void         Summary(Float_t &rt, Float_t &cp);
R__EXTERN TBenchmark  *gBenchmark;
```

code

```
gBenchmark->Start("tree");
gBenchmark->Show("tree");
//输出格式如下：tree       : Real Time =   0.55 seconds Cpu Time =
  0.17 seconds
```

TBranch

继承 TNamed , TAttFill

A TTree object is a list of TBranchs.
A TBranch describes the branch data structure and supports :

- the list of TBaskets (branch buffers) associated to this branch.
- the list of TLeaves (branch description)

class

```
   TBranch();
   TBranch(TTree *tree, const char *name, void *address, const c
har *leaflist, Int_t basketsize=32000, Int_t compress=-1);
/// Create a Branch as a child of a Tree
///       * address is the address of the first item of a struct
ure
///         or the address of a pointer to an object (see exampl
e in TTree.cxx).
///       * leaflist is the concatenation of all the variable na
mes and types
///         separated by a colon character :
///         The variable name and the variable type are separate
d by a
///         slash (/). The variable type must be 1 character. (C
haracters
///         after the first are legal and will be appended to th
e visible
///         name of the leaf, but have no effect.) If no type is
 given, the
///         type of the variable is assumed to be the same as th
e previous
///         variable. If the first variable does not have a type
, it is
///         assumed of type F by default. The list of currently
supported
```

```
///         types is given below:
///            - C : a character string terminated by the 0 char
acter
///            - B : an 8 bit signed integer (Char_t)
///            - b : an 8 bit unsigned integer (UChar_t)
///            - S : a 16 bit signed integer (Short_t)
///            - s : a 16 bit unsigned integer (UShort_t)
///            - I : a 32 bit signed integer (Int_t)
///            - i : a 32 bit unsigned integer (UInt_t)
///            - F : a 32 bit floating point (Float_t)
///            - D : a 64 bit floating point (Double_t)
///            - L : a 64 bit signed integer (Long64_t)
///            - l : a 64 bit unsigned integer (ULong64_t)
///            - O : [the letter 'o', not a zero] a boolean (Boo
l_t)
///         Arrays of values are supported with the following sy
ntax:
///         - If leaf name has the form var[nelem], where nelem
is alphanumeric, then
///              if nelem is a leaf name, it is used as the vari
able size of the array,
///              otherwise return 0.
///              The leaf referred to by nelem **MUST** be an in
t (/I),
///         - If leaf name has the form var[nelem], where nelem
is a non-negative integers, then
///              it is used as the fixed size of the array.
///         - If leaf name has the form of a multi dimension arr
ay (e.g. var[nelem][nelem2])
///              where nelem and nelem2 are non-negative integer
s) then
///              it is used as a 2 dimensional array of fixed si
ze.
///         - Any of other form is not supported.
///    Note that the TTree will assume that all the item are con
tiguous in memory.
///    On some platform, this is not always true of the member o
f a struct or a class,
///    due to padding and alignment.  Sorting your data member i
n order of decreasing
```

```
///    sizeof usually leads to their being contiguous in memory.
///       * bufsize is the buffer size in bytes for this branch
///         The default value is 32000 bytes and should be ok fo
r most cases.
///         You can specify a larger value (e.g. 256000) if your
 Tree is not split
///         and each entry is large (Megabytes)
///         A small value for bufsize is optimum if you intend t
o access
///         the entries in the Tree randomly and your Tree is in
 split mode.
///   Note that in case the data type is an object, this branch
can contain
///   only this object.
///    Note that this function is invoked by TTree::Branch

   TBranch(TBranch *parent, const char *name, void *address, con
st char *leaflist, Int_t basketsize=32000, Int_t compress=-1);
   virtual ~TBranch();

   virtual void      AddBasket(TBasket &b, Bool_t ondisk, Long64
_t startEntry);/// Add the basket to this branch.
   virtual void      AddLastBasket(Long64_t startEntry);/// Add
the start entry of the write basket (not yet created)
   virtual void      Browse(TBrowser *b);/// Browser interface.
   virtual void      DeleteBaskets(Option_t* option="");
 /// Loop on all branch baskets. If the file where branch buffer
s reside is
 /// writable, free the disk space associated to the baskets of
the branch,
 /// then call Reset(). If the option contains "all", delete als
o the baskets
 /// for the subbranches.
 /// The branch is reset.
 /// NOTE that this function must be used with extreme care. Del
eting branch baskets
 /// fragments the file and may introduce inefficiencies when ad
ding new entries
 /// in the Tree or later on when reading the Tree.
```

```
   virtual void      DropBaskets(Option_t *option = "");
/// Loop on all branch baskets. Drop all baskets from memory exc
ept readbasket.
/// If the option contains "all", drop all baskets including
/// read- and write-baskets (unless they are not stored individu
ally on disk).
/// The option "all" also lead to DropBaskets being called on th
e sub-branches.

           void      ExpandBasketArrays();
/// Increase BasketEntry buffer of a minimum of 10 locations
/// and a maximum of 50 per cent of current size.

   virtual Int_t     Fill();
/// Loop on all leaves of this branch to fill Basket buffer.
/// The function returns the number of bytes committed to the me
mory basket.
/// If a write error occurs, the number of bytes returned is -1.
/// If no data are written, because e.g. the branch is disabled,
/// the number of bytes returned is 0.

   virtual TBranch  *FindBranch(const char *name);/// Find the i
mmediate sub-branch with passed name.
   virtual TLeaf    *FindLeaf(const char *name);/// Find the lea
f corresponding to the name 'searchname'.
           Int_t     FlushBaskets();
/// Flush to disk all the baskets of this branch and any of subb
ranches.
/// Return the number of bytes written or -1 in case of write er
ror.

           Int_t     FlushOneBasket(UInt_t which);
/// If we have a write basket in memory and it contains some ent
ries and
/// has not yet been written to disk, we write it and delete it
from memory.
/// Return the number of bytes written;

   virtual char     *GetAddress() const {return fAddress;}
           TBasket  *GetBasket(Int_t basket);/// Return pointer
```

```
to basket basketnumber in this Branch
           Int_t    *GetBasketBytes() const {return fBasketBytes
;}
           Long64_t *GetBasketEntry() const {return fBasketEntry
;}
   virtual Long64_t  GetBasketSeek(Int_t basket) const;/// Retur
n address of basket in the file
   virtual Int_t     GetBasketSize() const {return fBasketSize;}
   virtual TList    *GetBrowsables();
/// Returns (and, if 0, creates) browsable objects for this bran
ch
/// See TVirtualBranchBrowsable::FillListOfBrowsables.

   virtual const char* GetClassName() const;
           Int_t     GetCompressionAlgorithm() const;
           Int_t     GetCompressionLevel() const;
           Int_t     GetCompressionSettings() const;
   TDirectory       *GetDirectory() const {return fDirectory;}
   virtual Int_t     GetEntry(Long64_t entry=0, Int_t getall = 0)
;
/// Read all leaves of entry and return total number of bytes re
ad.
/// The input argument "entry" is the entry number in the curren
t tree.
/// In case of a TChain, the entry number in the current Tree mu
st be found
/// before calling this function.
/// The function returns the number of bytes read from the input
 buffer.
/// If entry does not exist, the function returns 0.
/// If an I/O error occurs, the function returns -1.
/// See IMPORTANT REMARKS in TTree::GetEntry.

   virtual Int_t     GetEntryExport(Long64_t entry, Int_t getall
, TClonesArray *list, Int_t n);
/// Read all leaves of an entry and export buffers to real objec
ts in a TClonesArray list.
/// Returns total number of bytes read.

           Int_t     GetEntryOffsetLen() const { return fEntryOf
```

```
fsetLen; }
           Int_t     GetEvent(Long64_t entry=0) {return GetEntry
(entry);}
   const char       *GetIconName() const;/// Return icon name de
pending on type of branch.
   virtual Int_t     GetExpectedType(TClass *&clptr,EDataType &t
ype);
/// Fill expectedClass and expectedType with information on the
data type of the
/// object/values contained in this branch (and thus the type of
 pointers
/// expected to be passed to Set[Branch]Address
/// return 0 in case of success and > 0 in case of failure.

   virtual TLeaf    *GetLeaf(const char *name) const;/// Return
pointer to the 1st Leaf named name in thisBranch
   virtual TFile    *GetFile(Int_t mode=0);
/// Return pointer to the file where branch buffers reside, retu
rns 0
/// in case branch buffers reside in the same file as tree heade
r.
/// If mode is 1 the branch buffer file is recreated.

   const char       *GetFileName()    const {return fFileName.Da
ta();}
           Int_t     GetOffset()      const {return fOffset;}
           Int_t     GetReadBasket()  const {return fReadBasket;
}
           Long64_t  GetReadEntry()   const {return fReadEntry;}
           Int_t     GetWriteBasket() const {return fWriteBasket
;}
           Long64_t  GetTotalSize(Option_t *option="")   const;
/// Return total number of bytes in the branch (including curren
t buffer)

           Long64_t  GetTotBytes(Option_t *option="")    const;
/// Return total number of bytes in the branch (excluding curren
t buffer)
/// if option ="*" includes all sub-branches of this branch too
```

```
            Long64_t  GetZipBytes(Option_t *option="")    const;
/// Return total number of zip bytes in the branch
/// if option ="*" includes all sub-branches of this branch too

            Long64_t  GetEntryNumber() const {return fEntryNumber
;}
            Long64_t  GetFirstEntry()  const {return fFirstEntry;
 }
          TObjArray  *GetListOfBaskets()  {return &fBaskets;}
          TObjArray  *GetListOfBranches() {return &fBranches;}
          TObjArray  *GetListOfLeaves()   {return &fLeaves;}
            Int_t     GetMaxBaskets()  const  {return fMaxBaskets
;}
            Int_t     GetNleaves()     const {return fNleaves;}
            Int_t     GetSplitLevel()  const {return fSplitLevel;
}
            Long64_t  GetEntries()     const {return fEntries;}
            TTree    *GetTree()        const {return fTree;}
   virtual Int_t     GetRow(Int_t row);
/// Return all elements of one row unpacked in internal array fV
alues
/// [Actually just returns 1 (?)]

   virtual Bool_t    GetMakeClass() const;
/// Return whether this branch is in a mode where the object are
 decomposed
/// or not (Also known as MakeClass mode).

   TBranch          *GetMother() const;/// Get our top-level par
ent branch in the tree.
   TBranch          *GetSubBranch(const TBranch *br) const;
/// Find the parent branch of child.
/// Return 0 if child is not in this branch hierarchy.

   Bool_t            IsAutoDelete() const;/// Return kTRUE if an
 existing object in a TBranchObject must be deleted.
   Bool_t            IsFolder() const;/// Return kTRUE if more t
han one leaf or browsables, kFALSE otherwise.
   virtual void      KeepCircular(Long64_t maxEntries);/// keep
a maximum of fMaxEntries in memory
```

```
   virtual Int_t     LoadBaskets();
///  Baskets associated to this branch are forced to be in memor
y.
///  You can call TTree::SetMaxVirtualSize(maxmemory) to instruct

///  the system that the total size of the imported baskets does
 not
///  exceed maxmemory bytes.
///  The function returns the number of baskets that have been p
ut in memory.
///  This method may be called to force all baskets of one or mo
re branches
///  in memory when random access to entries in this branch is r
equired.
///  See also TTree::LoadBaskets to load all baskets of all bran
ches in memory.

   virtual void      Print(Option_t *option="") const;/// Print
TBranch parameters

   virtual void      ReadBasket(TBuffer &b);
   virtual void      Refresh(TBranch *b);
/// Refresh this branch using new information in b
/// This function is called by TTree::Refresh

   virtual void      Reset(Option_t *option="");
/// Reset a Branch.
/// - Existing buffers are deleted.
/// - Entries, max and min are reset.

   virtual void      ResetAfterMerge(TFileMergeInfo *);
/// Reset a Branch.
/// - Existing buffers are deleted.
/// - Entries, max and min are reset.

   virtual void      ResetAddress();/// Reset the address of the
 branch.

   virtual void      ResetReadEntry() {fReadEntry = -1;}
```

```
   virtual void      SetAddress(void *add);/// Set address of th
is branch.

   virtual void      SetObject(void *objadd);/// Set object this
 branch is pointing to.
   virtual void      SetAutoDelete(Bool_t autodel=kTRUE);
/// Set the automatic delete bit.
/// This bit is used by TBranchObject::ReadBasket to decide if a
n object
/// referenced by a TBranchObject must be deleted or not before
reading
/// a new entry.
/// If autodel is kTRUE, this existing object will be deleted, a
 new object
/// created by the default constructor, then read from disk by t
he streamer.
/// If autodel is kFALSE, the existing object is not deleted.  R
oot assumes
/// that the user is taking care of deleting any internal object
 or array
/// (this can be done in the streamer).

   virtual void      SetBasketSize(Int_t buffsize);
/// Set the basket size
/// The function makes sure that the basket size is greater than
 fEntryOffsetlen

   virtual void      SetBufferAddress(TBuffer *entryBuffer);
/// Set address of this branch directly from a TBuffer to avoid
streaming.
/// Note: We do not take ownership of the buffer.

   void              SetCompressionAlgorithm(Int_t algorithm=0);
/// Set compression algorithm.
   void              SetCompressionLevel(Int_t level=1);/// Set
compression level.
   void              SetCompressionSettings(Int_t settings=1);//
/ Set compression settings.
   virtual void      SetEntries(Long64_t entries);/// Set the nu
mber of entries in this branch.
```

```
   virtual void      SetEntryOffsetLen(Int_t len, Bool_t updateS
ubBranches = kFALSE);
/// Update the default value for the branch's fEntryOffsetLen if
 and only if
/// it was already non zero (and the new value is not zero)
/// If updateExisting is true, also update all the existing bran
ches.

   virtual void      SetFirstEntry( Long64_t entry );
///set the first entry number (case of TBranchSTL)

   virtual void      SetFile(TFile *file=0);
/// Set file where this branch writes/reads its buffers.
/// By default the branch buffers reside in the file where the
/// Tree was created.
/// If the file name where the tree was created is an absolute
/// path name or an URL (e.g. /castor/... or root://host/...)
/// and if the fname is not an absolute path name or an URL then
/// the path of the tree file is prepended to fname to make the
/// branch file relative to the tree file. In this case one can
/// move the tree + all branch files to a different location in
/// the file system and still access the branch files.
/// The ROOT file will be connected only when necessary.
/// If called by TBranch::Fill (via TBasket::WriteFile), the file

/// will be created with the option "recreate".
/// If called by TBranch::GetEntry (via TBranch::GetBasket), the
 file
/// will be opened in read mode.
/// To open a file in "update" mode or with a certain compression

/// level, use TBranch::SetFile(TFile *file).

   virtual void      SetFile(const char *filename);
/// Set file where this branch writes/reads its buffers.
/// By default the branch buffers reside in the file where the
/// Tree was created.
/// If the file name where the tree was created is an absolute
/// path name or an URL (e.g. /castor/... or root://host/...)
/// and if the fname is not an absolute path name or an URL then
```

```
/// the path of the tree file is prepended to fname to make the
/// branch file relative to the tree file. In this case one can
/// move the tree + all branch files to a different location in
/// the file system and still access the branch files.
/// The ROOT file will be connected only when necessary.
/// If called by TBranch::Fill (via TBasket::WriteFile), the file

/// will be created with the option "recreate".
/// If called by TBranch::GetEntry (via TBranch::GetBasket), the
 file
/// will be opened in read mode.
/// To open a file in "update" mode or with a certain compression

/// level, use TBranch::SetFile(TFile *file).

   virtual Bool_t    SetMakeClass(Bool_t decomposeObj = kTRUE);
/// Set the branch in a mode where the object are decomposed
/// (Also known as MakeClass mode).
/// Return whether the setting was possible (it is not possible
for
/// TBranch and TBranchObject).

   virtual void      SetOffset(Int_t offset=0) {fOffset=offset;}
   virtual void      SetStatus(Bool_t status=1);/// Set branch s
tatus to Process or DoNotProcess.
   virtual void      SetTree(TTree *tree) { fTree = tree;}
   virtual void      SetupAddresses();
/// If the branch address is not set,  we set all addresses star
ting with
/// the top level parent branch.

   virtual void      UpdateAddress() {;}
   virtual void      UpdateFile();
/// Refresh the value of fDirectory (i.e. where this branch writ
es/reads its buffers)
/// with the current value of fTree->GetCurrentFile unless this
branch has been
/// redirected to a different file.  Also update the sub-branche
s.
```

```
   static  void      ResetCount();/// Static function resetting
fgCount
```

code

```
TChain* chain = ...;
Long64_t localEntry = chain->LoadTree(entry);
branch->GetEntry(localEntry);
```

example

TBuffer

继承 TObject

Buffer base class used for serializing objects.

class

```
   enum EMode { kRead = 0, kWrite = 1 };
   enum { kIsOwner = BIT(16) };                        //if set
TBuffer owns fBuffer
   enum { kCannotHandleMemberWiseStreaming = BIT(17)}; //if set
TClonesArray should not use member wise streaming
   enum { kInitialSize = 1024, kMinimalSize = 128 };

   TBuffer(EMode mode);
   TBuffer(EMode mode, Int_t bufsiz);
   TBuffer(EMode mode, Int_t bufsiz, void *buf, Bool_t adopt = k
TRUE, ReAllocCharFun_t reallocfunc = 0);
   virtual ~TBuffer();

   Int_t    GetBufferVersion() const { return fVersion; }
   Bool_t   IsReading() const { return (fMode & kWrite) == 0; }
   Bool_t   IsWriting() const { return (fMode & kWrite) != 0; }
   void     SetReadMode();
   void     SetWriteMode();
   void     SetBuffer(void *buf, UInt_t bufsiz = 0, Bool_t adopt
 = kTRUE, ReAllocCharFun_t reallocfunc = 0);
   ReAllocCharFun_t GetReAllocFunc() const;
   void     SetReAllocFunc(ReAllocCharFun_t reallocfunc = 0);
   void     SetBufferOffset(Int_t offset = 0) { fBufCur = fBuffe
r+offset; }
   void     SetParent(TObject *parent);
   TObject *GetParent()  const;
   char    *Buffer()     const { return fBuffer; }
   Int_t    BufferSize() const { return fBufSize; }
   void     DetachBuffer() { fBuffer = 0; }
```

```
   Int_t    Length()     const { return (Int_t)(fBufCur - fBuffe
r); }
   void     Expand(Int_t newsize, Bool_t copy = kTRUE);  // expa
nd buffer to newsize
   void     AutoExpand(Int_t size_needed);  // expand buffer to
newsize

   virtual Bool_t     CheckObject(const TObject *obj) = 0;
   virtual Bool_t     CheckObject(const void *obj, const TClass
*ptrClass) = 0;

   virtual Int_t      ReadBuf(void *buf, Int_t max) = 0;
   virtual void       WriteBuf(const void *buf, Int_t max) = 0;

   virtual char      *ReadString(char *s, Int_t max) = 0;
   virtual void       WriteString(const char *s) = 0;

   virtual Int_t      GetVersionOwner() const  = 0;
   virtual Int_t      GetMapCount() const  = 0;
   virtual void       GetMappedObject(UInt_t tag, void* &ptr, TC
lass* &ClassPtr) const = 0;
   virtual void       MapObject(const TObject *obj, UInt_t offse
t = 1) = 0;
   virtual void       MapObject(const void *obj, const TClass *c
l, UInt_t offset = 1) = 0;
   virtual void       Reset() = 0;
   virtual void       InitMap() = 0;
   virtual void       ResetMap() = 0;
   virtual void       SetReadParam(Int_t mapsize) = 0;
   virtual void       SetWriteParam(Int_t mapsize) = 0;

   virtual Int_t      CheckByteCount(UInt_t startpos, UInt_t bcn
t, const TClass *clss) = 0;
   virtual Int_t      CheckByteCount(UInt_t startpos, UInt_t bcn
t, const char *classname) = 0;
   virtual void       SetByteCount(UInt_t cntpos, Bool_t packInV
ersion = kFALSE)= 0;

   virtual void       SkipVersion(const TClass *cl = 0) = 0;
   virtual Version_t  ReadVersion(UInt_t *start = 0, UInt_t *bcn
```

```
t = 0, const TClass *cl = 0) = 0;
   virtual Version_t  ReadVersionNoCheckSum(UInt_t *start = 0, U
Int_t *bcnt = 0) = 0;
   virtual Version_t  ReadVersionForMemberWise(const TClass *cl
= 0) = 0;
   virtual UInt_t     WriteVersion(const TClass *cl, Bool_t useB
cnt = kFALSE) = 0;
   virtual UInt_t     WriteVersionMemberWise(const TClass *cl, B
ool_t useBcnt = kFALSE) = 0;

   virtual void      *ReadObjectAny(const TClass* cast) = 0;
   virtual void       SkipObjectAny() = 0;

   virtual void       TagStreamerInfo(TVirtualStreamerInfo* info)
 = 0;
   virtual void       IncrementLevel(TVirtualStreamerInfo* info)
= 0;
   virtual void       SetStreamerElementNumber(TStreamerElement
*elem, Int_t comp_type) = 0;
   virtual void       DecrementLevel(TVirtualStreamerInfo*) = 0;

   virtual void       ClassBegin(const TClass*, Version_t = -1)
= 0;
   virtual void       ClassEnd(const TClass*) = 0;
   virtual void       ClassMember(const char*, const char* = 0,
Int_t = -1, Int_t = -1) = 0;
   virtual TVirtualStreamerInfo *GetInfo() = 0;

   virtual TVirtualArray *PeekDataCache() const;
   virtual TVirtualArray *PopDataCache();
   virtual void           PushDataCache(TVirtualArray *);

   virtual TClass    *ReadClass(const TClass *cl = 0, UInt_t *ob
jTag = 0) = 0;
   virtual void       WriteClass(const TClass *cl) = 0;

   virtual TObject   *ReadObject(const TClass *cl) = 0;
   virtual void       WriteObject(const TObject *obj) = 0;

   virtual Int_t      WriteObjectAny(const void *obj, const TCla
```

```
ss *ptrClass) = 0;

   virtual UShort_t   GetPidOffset() const  = 0;
   virtual void       SetPidOffset(UShort_t offset) = 0;
   virtual Int_t      GetBufferDisplacement() const  = 0;
   virtual void       SetBufferDisplacement() = 0;
   virtual void       SetBufferDisplacement(Int_t skipped) = 0;

   // basic types and arrays of basic types
   virtual   void     ReadFloat16 (Float_t *f, TStreamerElement
*ele=0) = 0;
   virtual   void     WriteFloat16(Float_t *f, TStreamerElement
*ele=0) = 0;
   virtual   void     ReadDouble32 (Double_t *d, TStreamerElemen
t *ele=0) = 0;
   virtual   void     WriteDouble32(Double_t *d, TStreamerElemen
t *ele=0) = 0;
   virtual   void     ReadWithFactor(Float_t *ptr, Double_t fact
or, Double_t minvalue) = 0;
   virtual   void     ReadWithNbits(Float_t *ptr, Int_t nbits) =
0;
   virtual   void     ReadWithFactor(Double_t *ptr, Double_t fac
tor, Double_t minvalue) = 0;
   virtual   void     ReadWithNbits(Double_t *ptr, Int_t nbits)
= 0;

   virtual   Int_t    ReadArray(Bool_t    *&b) = 0;
   virtual   Int_t    ReadArray(Char_t    *&c) = 0;
   virtual   Int_t    ReadArray(UChar_t   *&c) = 0;
   virtual   Int_t    ReadArray(Short_t   *&h) = 0;
   virtual   Int_t    ReadArray(UShort_t  *&h) = 0;
   virtual   Int_t    ReadArray(Int_t     *&i) = 0;
   virtual   Int_t    ReadArray(UInt_t    *&i) = 0;
   virtual   Int_t    ReadArray(Long_t    *&l) = 0;
   virtual   Int_t    ReadArray(ULong_t   *&l) = 0;
   virtual   Int_t    ReadArray(Long64_t  *&l) = 0;
   virtual   Int_t    ReadArray(ULong64_t *&l) = 0;
   virtual   Int_t    ReadArray(Float_t   *&f) = 0;
   virtual   Int_t    ReadArray(Double_t  *&d) = 0;
   virtual   Int_t    ReadArrayFloat16(Float_t *&f, TStreamerEle
```

```
ment *ele=0) = 0;
   virtual   Int_t    ReadArrayDouble32(Double_t *&d, TStreamerE
lement *ele=0) = 0;

   virtual   Int_t    ReadStaticArray(Bool_t    *b) = 0;
   virtual   Int_t    ReadStaticArray(Char_t    *c) = 0;
   virtual   Int_t    ReadStaticArray(UChar_t   *c) = 0;
   virtual   Int_t    ReadStaticArray(Short_t   *h) = 0;
   virtual   Int_t    ReadStaticArray(UShort_t  *h) = 0;
   virtual   Int_t    ReadStaticArray(Int_t     *i) = 0;
   virtual   Int_t    ReadStaticArray(UInt_t    *i) = 0;
   virtual   Int_t    ReadStaticArray(Long_t    *l) = 0;
   virtual   Int_t    ReadStaticArray(ULong_t   *l) = 0;
   virtual   Int_t    ReadStaticArray(Long64_t  *l) = 0;
   virtual   Int_t    ReadStaticArray(ULong64_t *l) = 0;
   virtual   Int_t    ReadStaticArray(Float_t   *f) = 0;
   virtual   Int_t    ReadStaticArray(Double_t  *d) = 0;
   virtual   Int_t    ReadStaticArrayFloat16(Float_t  *f, TStrea
merElement *ele=0) = 0;
   virtual   Int_t    ReadStaticArrayDouble32(Double_t  *d, TStr
eamerElement *ele=0) = 0;

   virtual   void     ReadFastArray(Bool_t    *b, Int_t n) = 0;
   virtual   void     ReadFastArray(Char_t    *c, Int_t n) = 0;
   virtual   void     ReadFastArrayString(Char_t *c, Int_t n) = 0
;
   virtual   void     ReadFastArray(UChar_t   *c, Int_t n) = 0;
   virtual   void     ReadFastArray(Short_t   *h, Int_t n) = 0;
   virtual   void     ReadFastArray(UShort_t  *h, Int_t n) = 0;
   virtual   void     ReadFastArray(Int_t     *i, Int_t n) = 0;
   virtual   void     ReadFastArray(UInt_t    *i, Int_t n) = 0;
   virtual   void     ReadFastArray(Long_t    *l, Int_t n) = 0;
   virtual   void     ReadFastArray(ULong_t   *l, Int_t n) = 0;
   virtual   void     ReadFastArray(Long64_t  *l, Int_t n) = 0;
   virtual   void     ReadFastArray(ULong64_t *l, Int_t n) = 0;
   virtual   void     ReadFastArray(Float_t   *f, Int_t n) = 0;
   virtual   void     ReadFastArray(Double_t  *d, Int_t n) = 0;
   virtual   void     ReadFastArrayFloat16(Float_t  *f, Int_t n,
 TStreamerElement *ele=0) = 0;
   virtual   void     ReadFastArrayDouble32(Double_t  *d, Int_t
```

```
n, TStreamerElement *ele=0) = 0;
   virtual   void     ReadFastArrayWithFactor(Float_t *ptr, Int_
t n, Double_t factor, Double_t minvalue) = 0;
   virtual   void     ReadFastArrayWithNbits(Float_t *ptr, Int_t
 n, Int_t nbits) = 0;
   virtual   void     ReadFastArrayWithFactor(Double_t *ptr, Int
_t n, Double_t factor, Double_t minvalue) = 0;
   virtual   void     ReadFastArrayWithNbits(Double_t *ptr, Int_
t n, Int_t nbits) = 0;
   virtual   void     ReadFastArray(void  *start , const TClass
*cl, Int_t n=1, TMemberStreamer *s=0, const TClass *onFileClass=0
) = 0;
   virtual   void     ReadFastArray(void **startp, const TClass
*cl, Int_t n=1, Bool_t isPreAlloc=kFALSE, TMemberStreamer *s=0,
const TClass *onFileClass=0) = 0;

   virtual   void     WriteArray(const Bool_t    *b, Int_t n) = 0
;
   virtual   void     WriteArray(const Char_t    *c, Int_t n) = 0
;
   virtual   void     WriteArray(const UChar_t   *c, Int_t n) = 0
;
   virtual   void     WriteArray(const Short_t   *h, Int_t n) = 0
;
   virtual   void     WriteArray(const UShort_t  *h, Int_t n) = 0
;
   virtual   void     WriteArray(const Int_t     *i, Int_t n) = 0
;
   virtual   void     WriteArray(const UInt_t    *i, Int_t n) = 0
;
   virtual   void     WriteArray(const Long_t    *l, Int_t n) = 0
;
   virtual   void     WriteArray(const ULong_t   *l, Int_t n) = 0
;
   virtual   void     WriteArray(const Long64_t  *l, Int_t n) = 0
;
   virtual   void     WriteArray(const ULong64_t *l, Int_t n) = 0
;
   virtual   void     WriteArray(const Float_t   *f, Int_t n) = 0
;
```

```
   virtual   void     WriteArray(const Double_t  *d, Int_t n) = 0
;
   virtual   void     WriteArrayFloat16(const Float_t  *f, Int_t
 n, TStreamerElement *ele=0) = 0;
   virtual   void     WriteArrayDouble32(const Double_t  *d, Int
_t n, TStreamerElement *ele=0) = 0;

   virtual   void     WriteFastArray(const Bool_t    *b, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Char_t    *c, Int_t n)
 = 0;
   virtual   void     WriteFastArrayString(const Char_t    *c, I
nt_t n) = 0;
   virtual   void     WriteFastArray(const UChar_t   *c, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Short_t   *h, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const UShort_t  *h, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Int_t     *i, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const UInt_t    *i, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Long_t    *l, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const ULong_t   *l, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Long64_t  *l, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const ULong64_t *l, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Float_t   *f, Int_t n)
 = 0;
   virtual   void     WriteFastArray(const Double_t  *d, Int_t n)
 = 0;
   virtual   void     WriteFastArrayFloat16(const Float_t  *f, I
nt_t n, TStreamerElement *ele=0) = 0;
   virtual   void     WriteFastArrayDouble32(const Double_t  *d,
 Int_t n, TStreamerElement *ele=0) = 0;
   virtual   void     WriteFastArray(void  *start,  const TClass
```

```
 *cl, Int_t n=1, TMemberStreamer *s=0) = 0;
   virtual   Int_t    WriteFastArray(void **startp, const TClass
 *cl, Int_t n=1, Bool_t isPreAlloc=kFALSE, TMemberStreamer *s=0)
= 0;

   virtual   void     StreamObject(void *obj, const type_info &t
ypeinfo, const TClass* onFileClass = 0 ) = 0;
   virtual   void     StreamObject(void *obj, const char *classN
ame, const TClass* onFileClass = 0 ) = 0;
   virtual   void     StreamObject(void *obj, const TClass *cl,
const TClass* onFileClass = 0 ) = 0;
   virtual   void     StreamObject(TObject *obj) = 0;

   virtual   void     ReadBool(Bool_t       &b) = 0;
   virtual   void     ReadChar(Char_t       &c) = 0;
   virtual   void     ReadUChar(UChar_t     &c) = 0;
   virtual   void     ReadShort(Short_t     &s) = 0;
   virtual   void     ReadUShort(UShort_t   &s) = 0;
   virtual   void     ReadInt(Int_t         &i) = 0;
   virtual   void     ReadUInt(UInt_t       &i) = 0;
   virtual   void     ReadLong(Long_t       &l) = 0;
   virtual   void     ReadULong(ULong_t     &l) = 0;
   virtual   void     ReadLong64(Long64_t   &l) = 0;
   virtual   void     ReadULong64(ULong64_t &l) = 0;
   virtual   void     ReadFloat(Float_t     &f) = 0;
   virtual   void     ReadDouble(Double_t   &d) = 0;
   virtual   void     ReadCharP(Char_t      *c) = 0;
   virtual   void     ReadTString(TString   &s) = 0;
   virtual   void     ReadStdString(std::string &s) = 0;

   virtual   void     WriteBool(Bool_t       b) = 0;
   virtual   void     WriteChar(Char_t       c) = 0;
   virtual   void     WriteUChar(UChar_t     c) = 0;
   virtual   void     WriteShort(Short_t     s) = 0;
   virtual   void     WriteUShort(UShort_t   s) = 0;
   virtual   void     WriteInt(Int_t         i) = 0;
   virtual   void     WriteUInt(UInt_t       i) = 0;
   virtual   void     WriteLong(Long_t       l) = 0;
   virtual   void     WriteULong(ULong_t     l) = 0;
   virtual   void     WriteLong64(Long64_t   l) = 0;
```

```
   virtual   void     WriteULong64(ULong64_t l) = 0;
   virtual   void     WriteFloat(Float_t     f) = 0;
   virtual   void     WriteDouble(Double_t   d) = 0;
   virtual   void     WriteCharP(const Char_t *c) = 0;
   virtual   void     WriteTString(const TString &s) = 0;
   virtual   void     WriteStdString(const std::string &s) = 0;

   // Special basic ROOT objects and collections
   virtual   TProcessID *GetLastProcessID(TRefTable *reftable) c
onst = 0;
   virtual   UInt_t      GetTRefExecId() = 0;
   virtual   TProcessID *ReadProcessID(UShort_t pidf) = 0;
   virtual   UShort_t    WriteProcessID(TProcessID *pid) = 0;

   // Utilities for TStreamerInfo
   virtual   void     ForceWriteInfo(TVirtualStreamerInfo *info,
 Bool_t force) = 0;
   virtual   void     ForceWriteInfoClones(TClonesArray *a) = 0;
   virtual   Int_t    ReadClones (TClonesArray *a, Int_t nobject
s, Version_t objvers) = 0;
   virtual   Int_t    WriteClones(TClonesArray *a, Int_t nobject
s) = 0;

   // Utilities for TClass
   virtual   Int_t    ReadClassEmulated(const TClass *cl, void *
object, const TClass *onfile_class = 0) = 0;
   virtual   Int_t    ReadClassBuffer(const TClass *cl, void *po
inter, const TClass *onfile_class = 0) = 0;
   virtual   Int_t    ReadClassBuffer(const TClass *cl, void *po
inter, Int_t version, UInt_t start, UInt_t count, const TClass *
onfile_class = 0) = 0;
   virtual   Int_t    WriteClassBuffer(const TClass *cl, void *p
ointer) = 0;

   // Utilites to streamer using sequences.
   virtual Int_t ApplySequence(const TStreamerInfoActions::TActi
onSequence &sequence, void *object) = 0;
   virtual Int_t ApplySequenceVecPtr(const TStreamerInfoActions:
:TActionSequence &sequence, void *start_collection, void *end_co
llection) = 0;
```

```
   virtual Int_t ApplySequence(const TStreamerInfoActions::TActi
onSequence &sequence, void *start_collection, void *end_collecti
on) = 0;

   static TClass *GetClass(const type_info &typeinfo);
   static TClass *GetClass(const char *className);
```

TBufferFile

TCanvas

继承 TPad

A Canvas is an area mapped to a window directly under the control of the display manager. A ROOT session may have several canvases open at any given time.

A Canvas may be subdivided into independent graphical areas: the **Pads**. A canvas has a default pad which has the name of the canvas itself.

class

```
   // TCanvas status bits
   enum {
      kShowEventStatus  = BIT(15),
      kAutoExec         = BIT(16),
      kMenuBar          = BIT(17),
      kShowToolBar      = BIT(18),
      kShowEditor       = BIT(19),
      kMoveOpaque       = BIT(20),
      kResizeOpaque     = BIT(21),
      kIsGrayscale      = BIT(22),
      kShowToolTips     = BIT(23)
   };

   TCanvas(Bool_t build=kTRUE);
   TCanvas(const char *name, const char *title="", Int_t form=1)
;
/// Create a new canvas with a predefined size form.
/// If form < 0  the menubar is not shown.
/// - form = 1    700x500 at 10,10 (set by TStyle::SetCanvasDefH
,W,X,Y)
/// - form = 2    500x500 at 20,20
/// - form = 3    500x500 at 30,30
/// - form = 4    500x500 at 40,40
/// - form = 5    500x500 at 50,50
/// If "name" starts with "gl" the canvas is ready to receive GL
```

```
 output.

   TCanvas(const char *name, const char *title, Int_t ww, Int_t
wh);
/// Create a new canvas at a random position.
/// \param[in] name    canvas name
/// \param[in] title   canvas title
/// \param[in] ww      is the canvas size in pixels along X
///                    (if ww < 0  the menubar is not shown)
/// \param[in] wh      is the canvas size in pixels along Y
/// If "name" starts with "gl" the canvas is ready to receive GL
 output.

   TCanvas(const char *name, const char *title, Int_t wtopx, Int
_t wtopy,
           Int_t ww, Int_t wh);
/// Create a new canvas.
/// \param[in] name         canvas name
/// \param[in] title        canvas title
/// \param[in] wtopx,wtopy  are the pixel coordinates of the top
 left corner of
///                         the canvas  (if wtopx < 0) the menub
ar is not shown)
/// \param[in] ww           is the canvas size in pixels along X
/// \param[in] wh           is the canvas size in pixels along Y
/// If "name" starts with "gl" the canvas is ready to receive GL
 output.

   TCanvas(const char *name, Int_t ww, Int_t wh, Int_t winid);
/// Create an embedded canvas, i.e. a canvas that is in a TGCanv
as widget
/// which is placed in a TGFrame. This ctor is only called via t
he
/// TRootEmbeddedCanvas class.
/// If "name" starts with "gl" the canvas is ready to receive GL
 output.

   virtual ~TCanvas();

   //-- used by friend TThread class
```

```
   void Constructor();
   void Constructor(const char *name, const char *title, Int_t f
orm);
/// Create a new canvas with a predefined size form.
/// If form < 0  the menubar is not shown.
/// - form = 1    700x500 at 10,10 (set by TStyle::SetCanvasDefH
,W,X,Y)
/// - form = 2    500x500 at 20,20
/// - form = 3    500x500 at 30,30
/// - form = 4    500x500 at 40,40
/// - form = 5    500x500 at 50,50

   void Constructor(const char *name, const char *title, Int_t w
w, Int_t wh);
/// Create a new canvas at a random position.
/// \param[in] name    canvas name
/// \param[in] title   canvas title
/// \param[in] ww      is the canvas size in pixels along X
///                    (if ww < 0  the menubar is not shown)
/// \param[in] wh      is the canvas size in pixels along Y

   void Constructor(const char *name, const char *title,
           Int_t wtopx, Int_t wtopy, Int_t ww, Int_t wh);
/// Create a new canvas.
/// \param[in] name         canvas name
/// \param[in] title        canvas title
/// \param[in] wtopx,wtopy  are the pixel coordinates of the top
 left corner of
///                         the canvas  (if wtopx < 0) the menub
ar is not shown)
/// \param[in] ww           is the canvas size in pixels along X
/// \param[in] wh           is the canvas size in pixels along Y

   void Destructor();/// Actual canvas destructor.

   TVirtualPad      *cd(Int_t subpadnumber=0);
/// Set current canvas & pad. Returns the new current pad,
/// or 0 in case of failure.
/// See TPad::cd() for an explanation of the parameter.
```

```
   virtual void      Browse(TBrowser *b);
   void              Clear(Option_t *option="");
// Remove all primitives from the canvas.
// If option "D" is specified, direct subpads are cleared but no
t deleted.
// This option is not recursive, i.e. pads in direct subpads are
 deleted.

   void              Close(Option_t *option="");/// Close canvas
. Delete window/pads data structure
   virtual void      Delete(Option_t * = "") { MayNotUse("Delete
()"); }
   void              DisconnectWidget();  // used by TCanvasImp
 /// Used by friend class TCanvasImp.
   virtual void      Draw(Option_t *option="");
/// Draw a canvas.
/// If a canvas with the name is already on the screen, the canv
as is repainted.
/// This function is useful when a canvas object has been saved
in a Root file.

   virtual TObject  *DrawClone(Option_t *option="") const; // *M
ENU*
/// Draw a clone of this canvas
/// A new canvas is created that is a clone of this canvas

   virtual TObject  *DrawClonePad(); // *MENU*
/// Draw a clone of this canvas into the current pad
/// In an interactive session, select the destination/current pad

/// with the middle mouse button, then point to the canvas area
to select
/// the canvas context menu item DrawClonePad.
/// Note that the original canvas may have subpads.

   virtual void      EditorBar();/// Get editor bar.
   void              EmbedInto(Int_t winid, Int_t ww, Int_t wh);
/// Embedded a canvas into a TRootEmbeddedCanvas. This method is
 only called
/// via TRootEmbeddedCanvas::AdoptCanvas.
```

```
   void              EnterLeave(TPad *prevSelPad, TObject *prevS
elObj);
/// Generate kMouseEnter and kMouseLeave events depending on the
 previously
/// selected object and the currently selected object. Does noth
ing if the
/// selected object does not change.

   void              FeedbackMode(Bool_t set);/// Turn rubberban
d feedback mode on or off.
   void              Flush();/// Flush canvas buffers.
   void              UseCurrentStyle(); // *MENU* /// Force a co
py of current style for all objects in canvas.
   void              ForceUpdate() { fCanvasImp->ForceUpdate();
}
   const char       *GetDISPLAY() const {return fDISPLAY.Data();
}
   TContextMenu     *GetContextMenu() const {return fContextMenu
;};
   Int_t             GetDoubleBuffer() const {return fDoubleBuff
er;}
   Int_t             GetEvent() const { return fEvent; }
   Int_t             GetEventX() const { return fEventX; }
   Int_t             GetEventY() const { return fEventY; }
   Color_t           GetHighLightColor() const { return fHighLig
htColor; }
   TVirtualPad      *GetPadSave() const { return fPadSave; }
   void              ClearPadSave() { fPadSave = 0; }
   TObject          *GetSelected() const {return fSelected;}
   TObject          *GetClickSelected() const {return fClickSele
cted;}
   Int_t             GetSelectedX() const {return fSelectedX;}
   Int_t             GetSelectedY() const {return fSelectedY;}
   Option_t         *GetSelectedOpt() const {return fSelectedOpt
.Data();}
   TVirtualPad      *GetSelectedPad() const { return fSelectedPa
d; }
   TVirtualPad      *GetClickSelectedPad() const { return fClick
SelectedPad; }
```

```
   Bool_t            GetShowEventStatus() const { return TestBit
(kShowEventStatus); }
   Bool_t            GetShowToolBar() const { return TestBit(kSh
owToolBar); }
   Bool_t            GetShowEditor() const { return TestBit(kSho
wEditor); }
   Bool_t            GetShowToolTips() const { return TestBit(kS
howToolTips); }
   Bool_t            GetAutoExec() const { return TestBit(kAutoE
xec); }
   Size_t            GetXsizeUser() const {return fXsizeUser;}
   Size_t            GetYsizeUser() const {return fYsizeUser;}
   Size_t            GetXsizeReal() const {return fXsizeReal;}
   Size_t            GetYsizeReal() const {return fYsizeReal;}
   Int_t             GetCanvasID() const {return fCanvasID;}
   TCanvasImp       *GetCanvasImp() const {return fCanvasImp;}
   Int_t             GetWindowTopX();/// Returns current top x p
osition of window on screen.
   Int_t             GetWindowTopY();/// Returns current top y p
osition of window on screen.
   UInt_t            GetWindowWidth() const { return fWindowWidt
h; }
   UInt_t            GetWindowHeight() const { return fWindowHei
ght; }
   UInt_t            GetWw() const { return fCw; }
   UInt_t            GetWh() const { return fCh; }
   virtual void      GetCanvasPar(Int_t &wtopx, Int_t &wtopy, UI
nt_t &ww, UInt_t &wh)
                     {wtopx=GetWindowTopX(); wtopy=fWindowTopY;
ww=fWindowWidth; wh=fWindowHeight;}
   virtual void      HandleInput(EEventType button, Int_t x, Int
_t y);
/// Handle Input Events.
/// Handle input events, like button up/down in current canvas.

   Bool_t            HasMenuBar() const { return TestBit(kMenuBa
r); }
   void              Iconify() { fCanvasImp->Iconify(); }
   Bool_t            IsBatch() const { return fBatch; }
   Bool_t            IsDrawn() { return fDrawn; }
```

```
   Bool_t            IsFolder() const;/// Is folder ?
   Bool_t            IsGrayscale();/// Check whether this canvas
 is to be drawn in grayscale mode.
   Bool_t            IsRetained() const { return fRetained; }
   virtual void      ls(Option_t *option="") const;/// List all
pads.
   void              MoveOpaque(Int_t set=1);
/// Set option to move objects/pads in a canvas.
///  - set = 1 (default) graphics objects are moved in opaque mo
de
///  - set = 0 only the outline of objects is drawn when moving
them
/// The option opaque produces the best effect. It requires howe
ver a
/// a reasonably fast workstation or response time.

   Bool_t            OpaqueMoving() const { return TestBit(kMove
Opaque); }
   Bool_t            OpaqueResizing() const { return TestBit(kRe
sizeOpaque); }
   virtual void      Paint(Option_t *option="");/// Paint canvas.

   virtual TPad     *Pick(Int_t px, Int_t py, TObjLink *&pickobj)
 { return TPad::Pick(px, py, pickobj); }
   virtual TPad     *Pick(Int_t px, Int_t py, TObject *prevSelOb
j);
/// Prepare for pick, call TPad::Pick() and when selected object
/// is different from previous then emit Picked() signal.

   virtual void      Picked(TPad *selpad, TObject *selected, Int
_t event);             // *SIGNAL*
/// Emit Picked() signal.

   virtual void      ProcessedEvent(Int_t event, Int_t x, Int_t
y, TObject *selected); // *SIGNAL*
/// Emit ProcessedEvent() signal.

   virtual void      Selected(TVirtualPad *pad, TObject *obj, In
t_t event);            // *SIGNAL*
/// Emit Selected() signal.
```

```
   virtual void      Cleared(TVirtualPad *pad);
                       // *SIGNAL*
/// Emit pad Cleared signal.

   virtual void      Closed();
                       // *SIGNAL*
/// Emit Closed signal.

   void              RaiseWindow() { fCanvasImp->RaiseWindow();
}
   void              ResetDrawn() { fDrawn=kFALSE; }
   virtual void      Resize(Option_t *option="");/// Recompute c
anvas parameters following a X11 Resize.
   void              ResizeOpaque(Int_t set=1);
/// Set option to resize objects/pads in a canvas.
///  - set = 1 (default) graphics objects are resized in opaque
mode
///  - set = 0 only the outline of objects is drawn when resizin
g them
/// The option opaque produces the best effect. It requires howe
ver a
/// a reasonably fast workstation or response time.

   void              SaveSource(const char *filename="", Option_
t *option="");
// Save primitives in this canvas as a C++ macro file.
// This function loops on all the canvas primitives and for each
 primitive
// calls the object SavePrimitive function.
// When outputing floating point numbers, the default precision
is 7 digits.
// The precision can be changed (via system.rootrc) by changing
the value
// of the environment variable "Canvas.SavePrecision"

   void              SavePrimitive(std::ostream &out, Option_t *
option = "");
   // Save primitives in this canvas in C++ macro file with GUI.
```

```
   virtual void      SetCursor(ECursor cursor);/// Set cursor.
   virtual void      SetDoubleBuffer(Int_t mode=1);/// Set Doubl
e Buffer On/Off.
   virtual void      SetFixedAspectRatio(Bool_t fixed = kTRUE);
// *TOGGLE*
/// Fix canvas aspect ratio to current value if fixed is true.

   void              SetGrayscale(Bool_t set = kTRUE); // *TOGGL
E* *GETTER=IsGrayscale
/// Set whether this canvas should be painted in grayscale, and
re-paint
/// it if necessary.

   void              SetWindowPosition(Int_t x, Int_t y) { fCanv
asImp->SetWindowPosition(x, y); }
   void              SetWindowSize(UInt_t ww, UInt_t wh) { fCanv
asImp->SetWindowSize(ww, wh); }
   void              SetCanvasSize(UInt_t ww, UInt_t wh); // *ME
NU*
/// Set Width and Height of canvas to ww and wh respectively. If
 ww and/or wh
/// are greater than the current canvas window a scroll bar is a
utomatically
/// generated. Use this function to zoom in a canvas and navigat
e via
/// the scroll bars. The Width and Height in this method are dif
ferent from those
/// given in the TCanvas constructors where these two dimension
include the size
/// of the window decoration whereas they do not in this method.

   void              SetHighLightColor(Color_t col) { fHighLight
Color = col; }
   void              SetSelected(TObject *obj);/// Set selected
canvas.
   void              SetClickSelected(TObject *obj) { fClickSele
cted = obj; }
   void              SetSelectedPad(TPad *pad) { fSelectedPad =
pad; }
   void              SetClickSelectedPad(TPad *pad) { fClickSele
```

```
ctedPad = pad; }
   void              Show() { fCanvasImp->Show(); }
   virtual void      Size(Float_t xsizeuser=0, Float_t ysizeuser=
0);
/// Set the canvas scale in centimeters.
/// This information is used by PostScript to set the page size.
/// \param[in] xsize   size of the canvas in centimeters along X
/// \param[in] ysize   size of the canvas in centimeters along Y
///  if xsize and ysize are not equal to 0, then the scale facto
rs will
///  be computed to keep the ratio ysize/xsize independently of
the canvas
///  size (parts of the physical canvas will be unused).
///  if xsize = 0 and ysize is not zero, then xsize will be comp
uted
///     to fit to the current canvas scale. If the canvas is res
ized,
///     a new value for xsize will be recomputed. In this case t
he aspect
///     ratio is not preserved.
///  if both xsize = 0 and ysize = 0, then the scaling is automa
tic.
///  the largest dimension will be allocated a size of 20 centim
eters.

   void              SetBatch(Bool_t batch=kTRUE);
/// Toggle batch mode. However, if the canvas is created without
 a window
/// then batch mode always stays set.

   static  void      SetFolder(Bool_t isfolder=kTRUE);
/// If isfolder=kTRUE, the canvas can be browsed like a folder
/// by default a canvas is not browsable.

   void              SetPadSave(TPad *pad) {fPadSave = pad;}
   void              SetRetained(Bool_t retained=kTRUE) { fRetai
ned=retained;}
   void              SetTitle(const char *title="");/// Set canv
as title.
   virtual void      ToggleEventStatus();/// Toggle event status
```

```
bar.
   virtual void      ToggleAutoExec();/// Toggle pad auto execut
ion of list of TExecs.
   virtual void      ToggleToolBar();/// Toggle toolbar.
   virtual void      ToggleEditor();/// Toggle editor.
   virtual void      ToggleToolTips();/// Toggle tooltip display.

   virtual void      Update();/// Update canvas pad buffers.  刷
新画板

   Bool_t              UseGL() const { return fUseGL; }
   void                SetSupportGL(Bool_t support) {fUseGL = su
pport;}
   TVirtualPadPainter *GetCanvasPainter();/// Access and (probab
ly) creation of pad painter.
   void                DeleteCanvasPainter();///assert on IsBatc
h() == false?

   static TCanvas   *MakeDefCanvas();/// Static function to buil
d a default canvas.
   static Bool_t     SupportAlpha();/// Static function returnin
g "true" if transparency is supported.
```

code

```
TCanvas *MyC = new TCanvas("MyC","Test canvas",1)      //新建画板
MyC->SetFillColor(42);                                 //设置画板背
景颜色
MyC->Divide(2,2);                                      //将画板分成
2*2四个区域
MyC->cd(1);                                            //指向第一个
区域
f1->Draw();                                            //在第一个区
域画图f1

MyC->SaveAs("");

TH1* hc = (TH1*)f1->Clone();                           //克隆直方图

//设置画板
TCanvas *c1 = new TCanvas("c1","画板标题在这",200,10,700,700);
TCanvas *c1 = new TCanvas("c1","画板标题在这",900,700);

c1->Divide(2,2);//将画板分成四份，2*2
c1->SetFillColor(40);//设置画板背景颜色


TCanvas *canv = new TCanvas("image", "ccc");
canv->ToggleEventStatus();
canv->SetRightMargin(0.2);
canv->SetLeftMargin(0.01);
canv->SetTopMargin(0.01);
canv->SetBottomMargin(0.01);
```

```
// At creation time, no matter if in interactive or batch mode,
the canvas size
// defines the size of the canvas window (including the size of
the window
// manager's decoration). To define precisely the graphics area
size of a canvas in
// the interactive mode, the following four lines of code should
 be used:

      Double_t w = 600;
      Double_t h = 600;
      TCanvas * c1 = new TCanvas("c", "c", w, h);
      c->SetWindowSize(w + (w - c->GetWw()), h + (h - c->GetWh()
));

// and in the batch mode simply do:

      c->SetCanvasSize(w,h);
```

example

```
Canvas *statsEditing() {
//     - how to remove a stat element from the stat box
//     - how to add a new one
//如何除去图上右上角box中的信息已经添加信息

   // Create and plot a test histogram with stats
   TCanvas *se = new TCanvas;
   TH1F *h = new TH1F("h","test",100,-3,3);
   h->FillRandom("gaus",3000);
   gStyle->SetOptStat();
   h->Draw();
   se->Update();

   // Retrieve the stat box
   TPaveStats *ps = (TPaveStats*)se->GetPrimitive("stats");
   ps->SetName("mystats");
   TList *list = ps->GetListOfLines();

   // Remove the RMS line
   TText *tconst = ps->GetLineWith("RMS");
   list->Remove(tconst);

   // Add a new line in the stat box.
   // Note that "=" is a control character
   TLatex *myt = new TLatex(0,0,"Test = 10");
   myt ->SetTextFont(42);
   myt ->SetTextSize(0.04);
   myt ->SetTextColor(kRed);
   list->Add(myt);

   // the following line is needed to avoid that the automatic r
edrawing of stats
   h->SetStats(0);

   se->Modified();
   return se;
}
```

3

TChain

class

继承 TTree

A chain is a collection of files containg TTree objects. When the chain is created, the first parameter is the default name for the Tree to be processed later on.

Enter a new element in the chain via the TChain::Add function. Once a chain is defined, one can use the normal TTree functions to Draw,Scan,etc.

Use TChain::SetBranchStatus to activate one or more branches for all the trees in the chain.

```
public:
   // TChain constants
   enum {
      kGlobalWeight   = BIT(15),
      kAutoDelete     = BIT(16),
      kProofUptodate  = BIT(17),
      kProofLite      = BIT(18)
   };

   // This used to be 1234567890, if user code hardcoded this nu
mber, the user code will need to change.
   static constexpr auto kBigNumber = TTree::kMaxEntries;

public:
   TChain();
   TChain(const char* name, const char* title = "");
   virtual ~TChain();

   virtual Int_t     Add(TChain* chain);
/// Add all files referenced by the passed chain to this chain.
/// The function returns the total number of files connected.
```

```
   virtual Int_t     Add(const char* name, Long64_t nentries = T
Tree::kMaxEntries);
/// Add a new file to this chain.
/// If tree_name is missing the chain name will be assumed.
/// Wildcard treatment is triggered by the any of the special ch
aracters []*?
/// which may be used in the file name, eg. specifying "xxx*.roo
t" adds
/// all files starting with xxx in the current file system direc
tory.
/// where "query" is to be interpreted by the remote server. Wil
dcards may be
/// supported in urls, depending on the protocol plugin and the
remote server.
/// http or https urls can contain a query identifier without tr
ee_name, but
/// generally urls can not be written with them because of ambig
uity with the
/// wildcard character. (Also see the documentaiton for TChain::
AddFile,
/// which does not support wildcards but allows the url to conta
in query)
/// NB. To add all the files of a TChain to a chain, use Add(TCh
ain *chain).
/// A. if nentries <= 0, the file is connected and the tree head
er read
///    in memory to get the number of entries.
/// B. if (nentries > 0, the file is not connected, nentries is
assumed to be
///    the number of entries in the file. In this case, no check
 is made that
///    the file exists and the Tree existing in the file. This s
econd mode
///    is interesting in case the number of entries in the file
is already stored
///    in a run data base for example.
/// C. if (nentries == TTree::kMaxEntries) (default), the file i
s not connected.
///    the number of entries in each file will be read only when
```

```
 the file
///    will need to be connected to read an entry.
///    This option is the default and very efficient if one proc
ess
///    the chain sequentially. Note that in case TChain::GetEntr
y(entry)
///    is called and entry refers to an entry in the 3rd file, f
or example,
///    this forces the Tree headers in the first and second file
///    to be read to find the number of entries in these files.
///    Note that if one calls TChain::GetEntriesFast() after hav
ing created
///    a chain with this default, GetEntriesFast will return TTr
ee::kMaxEntries!
///    TChain::GetEntries will force of the Tree headers in the
chain to be
///    read to read the number of entries in each Tree.
/// D. The TChain data structure
///    Each TChainElement has a name equal to the tree name of t
his TChain
///    and a title equal to the file name.
/// Return value:
/// - If nentries>0 (including the default of TTree::kMaxEntries
) and no
///   wildcarding is used, ALWAYS returns 1 without regard to wh
ether
///   the file exists or contains the correct tree.
/// - If wildcarding is used, regardless of the value of nentrie
s,
///   returns the number of files matching the name without rega
rd to
///   whether they contain the correct tree.
/// - If nentries<=0 and wildcarding is not used, return 1 if th
e file
///  exists and contains the correct tree and 0 otherwise.

   virtual Int_t     AddFile(const char* name, Long64_t nentries
 = TTree::kMaxEntries, const char* tname = "");
/// Add a new file to this chain.
/// Filename formats are similar to TChain::Add. Wildcards are n
```

```
ot
/// applied. urls may also contain query and fragment identifiers

/// where the tree name can be specified in the url fragment.
/// If tree_name is given as a part of the file name it is used
to
/// as the name of the tree to load from the file. Otherwise if
tname
/// argument is specified the chain will load the tree named tna
me from
/// the file, otherwise the original treename specified in the T
Chain
/// constructor will be used.
/// A. If nentries <= 0, the file is opened and the tree header
read
///    into memory to get the number of entries.
/// B. If nentries > 0, the file is not opened, and nentries is
assumed
///    to be the number of entries in the file. In this case, no
 check
///    is made that the file exists nor that the tree exists in
the file.
///    This second mode is interesting in case the number of ent
ries in
///    the file is already stored in a run database for example.
/// C. If nentries == TTree::kMaxEntries (default), the file is
not opened.
///    The number of entries in each file will be read only when
 the file
///    is opened to read an entry.  This option is the default a
nd very
///    efficient if one processes the chain sequentially.  Note
that in
///    case GetEntry(entry) is called and entry refers to an ent
ry in the
///    third file, for example, this forces the tree headers in
the first
///    and second file to be read to find the number of entries
in those
///    files.  Note that if one calls GetEntriesFast() after hav
```

```
ing created
///    a chain with this default, GetEntriesFast() will return T
Tree::kMaxEntries!
///    Using the GetEntries() function instead will force all of
 the tree
///    headers in the chain to be read to read the number of ent
ries in
///    each tree.
/// D. The TChain data structure
///    Each TChainElement has a name equal to the tree name of t
his TChain
///    and a title equal to the file name.
/// The function returns 1 if the file is successfully connected
, 0 otherwise.

   virtual Int_t     AddFileInfoList(TCollection* list, Long64_t
 nfiles = TTree::kMaxEntries);
/// Add all files referenced in the list to the chain. The objec
t type in the
/// list must be either TFileInfo or TObjString or TUrl .
/// The function return 1 if successful, 0 otherwise.

   virtual TFriendElement *AddFriend(const char* chainname, const
 char* dummy = "");
/// Add a TFriendElement to the list of friends of this chain.
/// A TChain has a list of friends similar to a tree (see TTree:
:AddFriend).
/// You can add a friend to a chain with the TChain::AddFriend m
ethod, and you
/// can retrieve the list of friends with TChain::GetListOfFrien
ds.
/// The parameter is the name of friend chain (the name of a cha
in is always
/// the name of the tree from which it was created).
/// The original chain has access to all variable in its friends.

/// We can use the TChain::Draw method as if the values in the f
riends were
/// in the original chain.
/// If the variable name is enough to uniquely identify the vari
```

```
able, you can
/// leave out the chain and/or branch name.
/// When a TChain::Draw is executed, an automatic call to TTree:
:AddFriend
/// connects the trees in the chain. When a chain is deleted, it
s friend
/// elements are also deleted.
/// The number of entries in the friend must be equal or greater
 to the number
/// of entries of the original chain. If the friend has fewer en
tries a warning
/// is given and the resulting histogram will have missing entri
es.
/// For additional information see TTree::AddFriend.

   virtual TFriendElement *AddFriend(const char* chainname, TFil
e* dummy);
/// Add the whole chain or tree as a friend of this chain.

   virtual TFriendElement *AddFriend(TTree* chain, const char* a
lias = "", Bool_t warn = kFALSE);
/// Add the whole chain or tree as a friend of this chain.

   virtual void      Browse(TBrowser*);/// Browse the contents o
f the chain.
   virtual void      CanDeleteRefs(Bool_t flag = kTRUE);
/// When closing a file during the chain processing, the file
/// may be closed with option "R" if flag is set to kTRUE.
/// by default flag is kTRUE.
/// When closing a file with option "R", all TProcessIDs referen
ced by this
/// file are deleted.
/// Calling TFile::Close("R") might be necessary in case one rea
ds a long list
/// of files having TRef, writing some of the referenced objects
 or TRef
/// to a new file. If the TRef or referenced objects of the file
 being closed
/// will not be referenced again, it is possible to minimize the
 size
```

```
/// of the TProcessID data structures in memory by forcing a del
ete of
/// the unused TProcessID.

   virtual void      CreatePackets();/// Initialize the packet d
escriptor string.
   virtual void      DirectoryAutoAdd(TDirectory *);/// Override
 the TTree::DirectoryAutoAdd behavior: we never auto add.
   virtual Long64_t  Draw(const char* varexp, const TCut& select
ion, Option_t* option = "", Long64_t nentries = kMaxEntries, Lon
g64_t firstentry = 0);
/// Draw expression varexp for selected entries.
/// Returns -1 in case of error or number of selected events in
case of success.
/// This function accepts TCut objects as arguments.
/// Useful to use the string operator +, example:
///    ntuple.Draw("x",cut1+cut2+cut3);

   virtual Long64_t  Draw(const char* varexp, const char* select
ion, Option_t* option = "", Long64_t nentries = kMaxEntries, Lon
g64_t firstentry = 0); // *MENU*
/// Process all entries in this chain and draw histogram corresp
onding to
/// expression varexp.
/// Returns -1 in case of error or number of selected events in
case of success.

   virtual void      Draw(Option_t* opt) { Draw(opt, "", "", kMa
xEntries, 0); }
   virtual Int_t     Fill() { MayNotUse("Fill()"); return -1; }
   virtual TBranch  *FindBranch(const char* name);/// See TTree:
:GetReadEntry().
   virtual TLeaf    *FindLeaf(const char* name);/// See TTree::G
etReadEntry().
   virtual TBranch  *GetBranch(const char* name);/// Return poin
ter to the branch name in the current tree.
   virtual Bool_t    GetBranchStatus(const char* branchname) con
st;/// See TTree::GetReadEntry().
   virtual Long64_t  GetCacheSize() const { return fTree ? fTree
->GetCacheSize() : fCacheSize; }
```

```
   virtual Long64_t  GetChainEntryNumber(Long64_t entry) const;
/// Return absolute entry number in the chain.
/// The input parameter entry is the entry number in
/// the current tree of this chain.

   virtual TClusterIterator GetClusterIterator(Long64_t firstent
ry);
/// Return an iterator over the cluster of baskets starting at f
irstentry.
/// This iterator is not yet supported for TChain object.

           Int_t     GetNtrees() const { return fNtrees; }
   virtual Long64_t  GetEntries() const;
/// Return the total number of entries in the chain.
/// In case the number of entries in each tree is not yet known,
/// the offset table is computed.

   virtual Long64_t  GetEntries(const char *sel) { return TTree:
:GetEntries(sel); }
   virtual Int_t     GetEntry(Long64_t entry=0, Int_t getall=0);
/// Get entry from the file to memory.
/// - getall = 0 : get only active branches
/// - getall = 1 : get all branches
/// Return the total number of bytes read,
/// 0 bytes read indicates a failure.

   virtual Long64_t  GetEntryNumber(Long64_t entry) const;
/// Return entry number corresponding to entry.
/// if no TEntryList set returns entry
/// else returns entry #entry from this entry list and
/// also computes the global entry number (loads all tree header
s)

   virtual Int_t     GetEntryWithIndex(Int_t major, Int_t minor=0
);
/// Return entry corresponding to major and minor number.
/// The function returns the total number of bytes read.
/// If the Tree has friend trees, the corresponding entry with
/// the index values (major,minor) is read. Note that the master
 Tree
```

```
/// and its friend may have different entry serial numbers corre
sponding
/// to (major,minor).

   TFile            *GetFile() const;
/// Return a pointer to the current file.
/// If no file is connected, the first file is automatically loa
ded.

   virtual TLeaf    *GetLeaf(const char* branchname, const char*
 leafname);/// Return a pointer to the leaf name in the current
tree.
   virtual TLeaf    *GetLeaf(const char* name);
/// Return a pointer to the leaf name in the current tree.

   virtual TObjArray *GetListOfBranches();
   //                Warning, GetListOfFiles returns the list of
 TChainElements (not the list of files)
/// Return a pointer to the list of branches of the current tree.

/// Warning: If there is no current TTree yet, this routine will
 open the
/// first in the chain.
/// Returns 0 on failure.

   //                see TChain::AddFile to see how to get the c
orresponding TFile objects
   TObjArray        *GetListOfFiles() const {return fFiles;}
   virtual TObjArray *GetListOfLeaves();
   virtual const char *GetAlias(const char *aliasName) const;///
 Returns the expanded value of the alias.  Search in the friends
 if any.
   virtual Double_t  GetMaximum(const char *columname);/// Retur
n maximum of column with name columname.
   virtual Double_t  GetMinimum(const char *columname);/// Retur
n minimum of column with name columname.
   virtual Int_t     GetNbranches();/// Return the number of bra
nches of the current tree. Warning: May set the current tree!
   virtual Long64_t  GetReadEntry() const;/// See TTree::GetRead
Entry().
```

```
   TList            *GetStatus() const { return fStatus; }
   virtual TTree    *GetTree() const { return fTree; }
   virtual Int_t     GetTreeNumber() const { return fTreeNumber;
 }
           Long64_t *GetTreeOffset() const { return fTreeOffset;
 }
           Int_t     GetTreeOffsetLen() const { return fTreeOffs
etLen; }
   virtual Double_t  GetWeight() const;
/// Return the chain weight.
/// By default the weight is the weight of the current tree.
/// However, if the weight has been set in TChain::SetWeight()
/// with the option "global", then that weight will be returned.
/// Warning: May set the current tree!

   virtual Int_t     LoadBaskets(Long64_t maxmemory);
/// Dummy function.
/// It could be implemented and load all baskets of all trees in
 the chain.
/// For the time being use TChain::Merge and TTree::LoadBasket
/// on the resulting tree.

   virtual Long64_t  LoadTree(Long64_t entry);
/// Find the tree which contains entry, and set it as the curren
t tree.
/// Returns the entry number in that tree.
/// The input argument entry is the entry serial number in the w
hole chain.
/// In case of error, LoadTree returns a negative number:
///   1. The chain is empty.
///   2. The requested entry number of less than zero or too lar
ge for the chain.
///       or too large for the large TTree.
///   3. The file corresponding to the entry could not be correc
tly open
///   4. The TChainElement corresponding to the entry is missing
 or
///       the TTree is missing from the file.
/// Note: This is the only routine which sets the value of fTree
 to
```

```
///       a non-zero pointer.

           void      Lookup(Bool_t force = kFALSE);
/// Check / locate the files in the chain.
/// By default only the files not yet looked up are checked.
/// Use force = kTRUE to check / re-check every file.

   virtual void      Loop(Option_t *option="", Long64_t nentries
=kMaxEntries, Long64_t firstentry=0); // *MENU* /// Loop on nent
ries of this chain starting at firstentry.  (NOT IMPLEMENTED)
   virtual void      ls(Option_t *option="") const;/// List the
chain.
   virtual Long64_t  Merge(const char *name, Option_t *option =
"");
/// Merge all the entries in the chain into a new tree in a new
file.
/// See important note in the following function Merge().
/// If the chain is expecting the input tree inside a directory,
/// this directory is NOT created by this routine.

   virtual Long64_t  Merge(TCollection *list, Option_t *option =
"");/// Merge all chains in the collection.  (NOT IMPLEMENTED)
   virtual Long64_t  Merge(TCollection *list, TFileMergeInfo *in
fo);/// Merge all chains in the collection.  (NOT IMPLEMENTED)
   virtual Long64_t  Merge(TFile *file, Int_t basketsize, Option
_t *option="");
/// Merge all the entries in the chain into a new tree in the cu
rrent file.
/// Note: The "file" parameter is *not* the file where the new
///       tree will be inserted.  The new tree is inserted into
///       gDirectory, which is usually the most recently opened
///       file, or the directory most recently cd()'d to.
/// If option = "C" is given, the compression level for all bran
ches
/// in the new Tree is set to the file compression level.  By de
fault,
/// the compression level of all branches is the original compre
ssion
/// level in the old trees.
/// If basketsize > 1000, the basket size for all branches of the
```

```
/// new tree will be set to basketsize.
/// If 'option' contains the word 'fast' the merge will be done
without
/// unzipping or unstreaming the baskets (i.e., a direct copy of
 the raw
/// bytes on disk).
/// When 'fast' is specified, 'option' can also contains a
/// sorting order for the baskets in the output file.
/// There is currently 3 supported sorting order:
///     SortBasketsByOffset (the default)
///     SortBasketsByBranch
///     SortBasketsByEntry
/// When using SortBasketsByOffset the baskets are written in
/// the output file in the same order as in the original file
/// (i.e. the basket are sorted on their offset in the original
/// file; Usually this also means that the baskets are sorted
/// on the index/number of the _last_ entry they contain)
/// When using SortBasketsByBranch all the baskets of each
/// individual branches are stored contiguously.  This tends to
/// optimize reading speed when reading a small number (1->5) of
/// branches, since all their baskets will be clustered together
/// instead of being spread across the file.  However it might
/// decrease the performance when reading more branches (or the
full
/// entry).
/// When using SortBasketsByEntry the baskets with the lowest
/// starting entry are written first.  (i.e. the baskets are
/// sorted on the index/number of the first entry they contain).
/// This means that on the file the baskets will be in the order
/// in which they will be needed when reading the whole tree
/// sequentially.
/// ## IMPORTANT Note 1: AUTOMATIC FILE OVERFLOW
/// When merging many files, it may happen that the resulting fi
le
/// reaches a size > TTree::fgMaxTreeSize (default = 1.9 GBytes).

/// In this case the current file is automatically closed and a
new
/// file started.  If the name of the merged file was "merged.ro
```

```
ot",
/// the subsequent files will be named "merged_1.root", "merged_
2.root",
/// etc.  fgMaxTreeSize may be modified via the static function
/// TTree::SetMaxTreeSize.
/// When in fast mode, the check and switch is only done in betw
een each
/// input file.
/// ## IMPORTANT Note 2: The output file is automatically closed
 and deleted.
/// This is required because in general the automatic file overf
low described
/// above may happen during the merge.
/// If only the current file is produced (the file passed as fir
st argument),
/// one can instruct Merge to not close and delete the file by s
pecifying
/// the option "keep".

   virtual void      Print(Option_t *option="") const;
/// Print the header information of each tree in the chain.
/// See TTree::Print for a list of options.

   virtual Long64_t  Process(const char *filename, Option_t *opt
ion="", Long64_t nentries=kMaxEntries, Long64_t firstentry=0); /
/ *MENU*
/// Process all entries in this chain, calling functions in file
name.
/// The return value is -1 in case of error and TSelector::GetSt
atus() in
/// in case of success.
/// See TTree::Process.

   virtual Long64_t  Process(TSelector* selector, Option_t* opti
on = "", Long64_t nentries = kMaxEntries, Long64_t firstentry = 0
);
/// Process this chain executing the code in selector.
/// The return value is -1 in case of error and TSelector::GetSt
atus() in
/// in case of success.
```

```
   virtual void      RecursiveRemove(TObject *obj);/// Make sure
 that obj (which is being deleted or will soon be) is no longer
referenced by this TTree.
   virtual void      RemoveFriend(TTree*);/// Remove a friend fr
om the list of friends.
   virtual void      Reset(Option_t *option="");/// Resets the s
tate of this chain.
   virtual void      ResetAfterMerge(TFileMergeInfo *);/// Reset
s the state of this chain after a merge (keep the customization
but forget the data).
   virtual void      ResetBranchAddress(TBranch *);/// Reset the
 addresses of the branch.
   virtual void      ResetBranchAddresses();/// Reset the addres
ses of the branches.
   virtual Long64_t  Scan(const char *varexp="", const char *sel
ection="", Option_t *option="", Long64_t nentries=kMaxEntries, L
ong64_t firstentry=0); // *MENU*
/// Loop on tree and print entries passing selection.
/// - If varexp is 0 (or "") then print only first 8 columns.
/// - If varexp = "*" print all columns.
/// - Otherwise a columns selection can be made using "var1:var2
:var3".
/// See TTreePlayer::Scan for more information.

   virtual void      SetAutoDelete(Bool_t autodel=kTRUE);
/// Set the global branch kAutoDelete bit.
/// When LoadTree loads a new Tree, the branches for which
/// the address is set will have the option AutoDelete set
/// For more details on AutoDelete, see TBranch::SetAutoDelete.

   virtual Int_t     SetBranchAddress(const char *bname,void *ad
d, TBranch **ptr = 0);
/// Set branch address.
/// \param[in] bname    is the name of a branch.
/// \param[in] add      is the address of the branch.
/// Note: See the comments in TBranchElement::SetAddress() for a
 more
/// detailed discussion of the meaning of the add parameter.
/// IMPORTANT REMARK:
```

```
/// In case TChain::SetBranchStatus is called, it must be called
/// BEFORE calling this function.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.

   virtual Int_t     SetBranchAddress(const char *bname,void *ad
d, TBranch **ptr, TClass *realClass, EDataType datatype, Bool_t
isptr);
/// Check if bname is already in the status list, and if not, cr
eate a TChainElement object and set its address.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.
/// Note: See the comments in TBranchElement::SetAddress() for a
 more
/// detailed discussion of the meaning of the add parameter.

   virtual Int_t     SetBranchAddress(const char *bname,void *ad
d, TClass *realClass, EDataType datatype, Bool_t isptr);
/// Check if bname is already in the status list, and if not, cr
eate a TChainElement object and set its address.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.
/// Note: See the comments in TBranchElement::SetAddress() for a
 more
/// detailed discussion of the meaning of the add parameter.

   template <class T> Int_t SetBranchAddress(const char *bname,
T **add, TBranch **ptr = 0) {
     return TTree::SetBranchAddress<T>(bname, add, ptr);
   }
#ifndef R__NO_CLASS_TEMPLATE_SPECIALIZATION
   // This can only be used when the template overload resolutio
n can distringuish between
   // T* and T**
   template <class T> Int_t SetBranchAddress(const char *bname,
T *add, TBranch **ptr = 0) {
     return TTree::SetBranchAddress<T>(bname, add, ptr);
   }
#endif
```

```
   virtual void      SetBranchStatus(const char *bname, Bool_t s
tatus=1, UInt_t *found=0);
/// Set branch status to Process or DoNotProcess
/// \param[in] bname     is the name of a branch. if bname="*",
apply to all branches.
/// \param[in] status    = 1  branch will be processed,
///                      = 0  branch will not be processed
///  See IMPORTANT REMARKS in TTree::SetBranchStatus and TChain:
:SetBranchAddress
///  If found is not 0, the number of branch(es) found matching
the regular
///  expression is returned in *found AND the error message 'unk
nown branch'
///  is suppressed.

   virtual Int_t     SetCacheSize(Long64_t cacheSize = -1);
   virtual void      SetDirectory(TDirectory *dir);
/// Remove reference to this chain from current directory and add

/// reference to new directory dir. dir can be 0 in which case t
he chain
/// does not belong to any directory.

   virtual void      SetEntryList(TEntryList *elist, Option_t *o
pt="");
/// Set the input entry list (processing the entries of the chai
n will then be
/// limited to the entries in the list)
/// This function finds correspondance between the sub-lists of
the TEntryList
/// and the trees of the TChain
/// By default (opt=""), both the file names of the chain elemen
ts and
/// the file names of the TEntryList sublists are expanded to fu
ll path name.
/// If opt = "ne", the file names are taken as they are and not
expanded

   virtual void      SetEntryListFile(const char *filename="", O
ption_t *opt="");
```

```
/// Set the input entry list (processing the entries of the chai
n will then be
/// limited to the entries in the list). This function creates a
 special kind
/// of entry list (TEntryListFromFile object) that loads lists,
corresponding
/// to the chain elements, one by one, so that only one list is
in memory at a time.
/// If there is an error opening one of the files, this file is
skipped and the next file is loaded
/// File naming convention:
/// - by default, filename_elist.root is used, where filename is
 the
///   name of the chain element
/// - xxx$xxx.root - $ sign is replaced by the name of the chain
 element
/// If the list name is not specified (by passing filename_elist
.root/listname to
/// the TChain::SetEntryList() function, the first object of cla
ss TEntryList
/// in the file is taken.
/// It is assumed, that there are as many list files, as there a
re elements in
/// the chain and they are in the same order

   virtual void      SetEventList(TEventList *evlist);
/// This function transfroms the given TEventList into a TEntryL
ist
/// NOTE, that this function loads all tree headers, because the
 entry numbers
/// in the TEventList are global and have to be recomputed, taki
ng into account
/// the number of entries in each tree.
/// The new TEntryList is owned by the TChain and gets deleted w
hen the chain
/// is deleted. This TEntryList is returned by GetEntryList() fu
nction, and after
/// GetEntryList() function is called, the TEntryList is not own
ed by the chain
/// any more and will not be deleted with it.
```

```
   virtual void      SetMakeClass(Int_t make) { TTree::SetMakeCl
ass(make); if (fTree) fTree->SetMakeClass(make);}
   virtual void      SetPacketSize(Int_t size = 100);/// Set num
ber of entries per packet for parallel root.
   virtual void      SetProof(Bool_t on = kTRUE, Bool_t refresh
= kFALSE, Bool_t gettreeheader = kFALSE);
/// Enable/Disable PROOF processing on the current default Proof
 (gProof).
/// "Draw" and "Processed" commands will be handled by PROOF.
/// The refresh and gettreeheader are meaningfull only if on ==
kTRUE.
/// If refresh is kTRUE the underlying fProofChain (chain proxy)
 is always
/// rebuilt (even if already existing).
/// If gettreeheader is kTRUE the header of the tree will be rea
d from the
/// PROOF cluster: this is only needed for browsing and should b
e used with
/// care because it may take a long time to execute.

   virtual void      SetWeight(Double_t w=1, Option_t *option="")
;
/// Set chain weight.
/// The weight is used by TTree::Draw to automatically weight ea
ch
/// selected entry in the resulting histogram.
/// For example the equivalent of
/// ~~~ {.cpp}
///     chain.Draw("x","w")
/// ~~~
/// is
/// ~~~ {.cpp}
///     chain.SetWeight(w,"global");
///     chain.Draw("x");
/// ~~~
/// By default the weight used will be the weight
/// of each Tree in the TChain. However, one can force the indiv
idual
/// weights to be ignored by specifying the option "global".
```

```
/// In this case, the TChain global weight will be used for all
Trees.

   virtual void      UseCache(Int_t maxCacheSize = 10, Int_t pag
eSize = 0);
```

code

```
///  Suppose we have 3 files f1.root, f2.root and f3.root. Each
file contains a TTree object named "T".
TChain ch("T");  creates a chain to process a Tree called "T"
ch.Add("f1.root");
ch.Add("f2.root");
ch.Add("f3.root");
ch.Draw("x");
```

```
/// Each TChainElement has a name equal to the tree name of this
 TChain and a title equal to the file name. So, to loop over the
 TFiles that have been added to this chain:

TObjArray *fileElements=chain->GetListOfFiles();
TIter next(fileElements);
TChainElement *chEl=0;
while (( chEl=(TChainElement*)next() )) {
   TFile f(chEl->GetTitle());
//   ... do something with f ...
}
```

```
/// Example using the file generated in $ROOTSYS/test/Event
/// merge two copies of Event.root

gSystem.Load("libEvent");
TChain ch("T");
ch.Add("Event1.root");
ch.Add("Event2.root");
ch.Merge("all.root");
```

```
/// If the chain is expecting the input tree inside a directory,
/// this directory is NOT created by this routine.
///
/// So if you do:

TChain ch("mydir/mytree");
ch.Merge("newfile.root");

/// The resulting file will not have subdirectories.  In order to

/// preserve the directory structure do the following instead:

TFile* file = TFile::Open("newfile.root", "RECREATE");
file->mkdir("mydir")->cd();
ch.Merge(file);

/// The function returns the total number of files produced.
/// To check that all files have been merged use something like:

if (newchain->GetEntries()!=oldchain->GetEntries()) {
   ... not all the file have been copied ...
}
```

```
/// This example has four chains each has 20 ROOT trees from 20
ROOT files.

TChain ch("t"); // a chain with 20 trees from 20 files
TChain ch1("t1");
TChain ch2("t2");
TChain ch3("t3");

/// Now we can add the friends to the first chain.

ch.AddFriend("t1")
ch.AddFriend("t2")
ch.AddFriend("t3")

/// For example, this generates a 3-d scatter plot of variable "
var" in the
/// TChain ch versus variable v1 in TChain t1 versus variable v2
 in TChain t2.

ch.Draw("var:t1.v1:t2.v2");
```

example

TCut

继承 TNamed

A specialized string object used in TTree selections.

class

```
   TCut();
   TCut(const char *title);
   TCut(const char *name, const char *title);
   TCut(const TCut &cut);
   virtual ~TCut();

   // Assignment
   TCut&    operator=(const char *rhs);
   TCut&    operator=(const TCut &rhs);
   TCut&    operator+=(const char *rhs);
   TCut&    operator+=(const TCut &rhs);
   TCut&    operator*=(const char *rhs);
   TCut&    operator*=(const TCut &rhs);

   // Comparison
   Bool_t   operator==(const char *rhs) const;
   Bool_t   operator==(const TCut &rhs) const;
   Bool_t   operator!=(const char *rhs) const;
   Bool_t   operator!=(const TCut &rhs) const;

   friend TCut operator+(const TCut &lhs, const char *rhs);
   friend TCut operator+(const char *lhs, const TCut &rhs);
   friend TCut operator+(const TCut &lhs, const TCut &rhs);
   friend TCut operator*(const TCut &lhs, const char *rhs);
   friend TCut operator*(const char *lhs, const TCut &rhs);
   friend TCut operator*(const TCut &lhs, const TCut &rhs);
// Preventing warnings with -Weffc++ in GCC since the overloadin
g of the && and || operators was a design choice.
#if (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL
```

```
__) >= 40600
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Weffc++"
#endif
   friend TCut operator&&(const TCut &lhs, const char *rhs);
   friend TCut operator&&(const char *lhs, const TCut &rhs);
   friend TCut operator&&(const TCut &lhs, const TCut &rhs);
   friend TCut operator||(const TCut &lhs, const char *rhs);
   friend TCut operator||(const char *lhs, const TCut &rhs);
   friend TCut operator||(const TCut &lhs, const TCut &rhs);
#if (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL
__) >= 40600
#pragma GCC diagnostic pop
#endif
   friend TCut operator!(const TCut &rhs);

   // Type conversion
   operator const char*() const { return GetTitle(); }
```

code

```
// A specialized string object used for TTree selections.
// A TCut object has a name and a title. It does not add any data

// members compared to a TNamed. It only add a set of operators
to
// facilitate logical string concatenation.
// Operators =, +=, +, *, !, &&, || overloaded.

    Root > TCut c1 = "x<1"
    Root > TCut c2 = "y<0"
    Root > TCut c3 = c1&&c2
    Root > ntuple.Draw("x", c1)
    Root > ntuple.Draw("x", c1||"x>0")
    Root > ntuple.Draw("x", c1&&c2)
    Root > ntuple.Draw("x", "(x+y)"*(c1&&c2))
```

example

TCutG

继承 TGraph

A Graphical cut.

A TCutG object is a closed polygon defining a closed region in a x,y plot. It can be created via the graphics editor option "CutG" or directly by invoking its constructor. The first and last points should be the same.

To create a TCutG via the graphics editor, use the left button to select the points building the contour of the cut. Click on the right button to close the TCutG. When it is created via the graphics editor, the TCutG object is named "CUTG". It is recommended to immediately change the name by using the context menu item "SetName". When the graphics editor is used, the names of the variables X,Y are automatically taken from the current pad title.

A Graphical cut may be drawn via TGraph::Draw. It can be edited like a normal TGraph.

A Graphical cut may be saved to a file via TCutG::Write.

class

```
   TCutG();/// TCutG default constructor.
   TCutG(const TCutG &cutg);/// TCutG copy constructor
   TCutG(const char *name, Int_t n);/// TCutG normal constructor.

   TCutG(const char *name, Int_t n, const Float_t *x, const Floa
t_t *y);/// TCutG normal constructor.
   TCutG(const char *name, Int_t n, const Double_t *x, const Dou
ble_t *y);/// TCutG normal constructor.
   virtual ~TCutG();

   TCutG &operator=(const TCutG &);
   virtual Double_t Area() const;
/// Compute the area inside this TCutG
```

```
/// The algorithm uses Stoke's theorem over the border of the cl
osed polygon.
/// Just as a reminder: Stoke's theorem reduces a surface integr
al
/// to a line integral over the border of the surface integral.

   virtual void     Center(Double_t &cx, Double_t &cy) const;
/// Compute the center x,y of this TCutG
/// The algorithm uses Stoke's theorem over the border of the cl
osed polygon.
/// Just as a reminder: Stoke's theorem reduces a surface integr
al
/// to a line integral over the border of the surface integral.

   TObject         *GetObjectX() const {return fObjectX;}
   TObject         *GetObjectY() const {return fObjectY;}
   const char      *GetVarX() const {return fVarX.Data();}
   const char      *GetVarY() const {return fVarY.Data();}
   virtual Double_t IntegralHist(TH2 *h, Option_t *option="") co
nst;
/// Compute the integral of 2-d histogram h for all bins inside
the cut
/// if option "width" is specified, the integral is the sum of
/// the bin contents multiplied by the bin width in x and in y.

   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");/// Save primitive as a C++ statement(s) on output s
tream out.
   virtual void     SetObjectX(TObject *obj);/// Set the X objec
t (and delete the previous one if any).
   virtual void     SetObjectY(TObject *obj);/// Set the Y objec
t (and delete the previous one if any).
   virtual void     SetVarX(const char *varx); // *MENU* /// Set
 X variable.
   virtual void     SetVarY(const char *vary); // *MENU* /// Set
 Y variable.
```

code

```
// Assume a TTree object T and:

Root > T.Draw("abs(fMomemtum)%fEtot")

// the TCutG members fVarX, fVary will be set to:

fVarx = fEtot
fVary = abs(fMomemtum)

// A graphical cut can be used in a TTree selection expression:

Root > T.Draw("fEtot","cutg1")

// where "cutg1" is the name of an existing graphical cut.

// Note that, as shown in the example above, a graphical cut may
 be used in a
// selection expression when drawing TTrees expressions of 1-d,
2-d or
// 3-dimensions. The expressions used in TTree::Draw can referen
ce the variables in
// the fVarX, fVarY of the graphical cut plus other variables.
```

```
// When the TCutG object is created by TTree::Draw, it is added
to the list of special objects in
// the main TROOT object pointed by gROOT. To retrieve a pointer
 to this object
// from the code or command line, do:

    TCutG *mycutg;
    mycutg = (TCutG*)gROOT->GetListOfSpecials()->FindObject("CUT
G")
    mycutg->SetName("mycutg");

// When the TCutG is not created via TTree::Draw, one must set t
he variable names
// corresponding to x,y if one wants to use the cut as input to
TTree::Draw,eg

    TCutG *cutg = new TCutG("mycut",5);
    cutg->SetVarX("y");
    cutg->SetVarY("x");
    cutg->SetPoint(0,-0.3586207,1.509534);
    cutg->SetPoint(1,-1.894181,-0.529661);
    cutg->SetPoint(2,0.07780173,-1.21822);
    cutg->SetPoint(3,-1.0375,-0.07944915);
    cutg->SetPoint(4,0.756681,0.1853814);
    cutg->SetPoint(5,-0.3586207,1.509534);

// Example of use of a TCutG in TTree::Draw:

    tree.Draw("x:y","mycutg && z>0 %% sqrt(x)>1")
```

example

```
//图像上数据筛选  曲线拟合
#include "TF2.h"
#include "TH2.h"
#include "TCutG.h"
#include "TMath.h"
#include "TCanvas.h"
```

```
#include "TStyle.h"
//+ Fitting a 2-D histogram (a variant)
// This tutorial illustrates :
//  - how to create a 2-d function
//  - fill a 2-d histogram randomly from this function
//  - fit the histogram
//  - display the fitted function on top of the histogram (lego-
plot)
//    using a surface plot in a sub-range of the histogram.
//      图像上数据筛选  曲线拟合
Double_t g2(Double_t *x, Double_t *par) {
   Double_t r1 = Double_t((x[0]-par[1])/par[2]);
   Double_t r2 = Double_t((x[1]-par[3])/par[4]);
   return par[0]*TMath::Exp(-0.5*(r1*r1+r2*r2));
}
Double_t fun2(Double_t *x, Double_t *par) {
   Double_t *p1 = &par[0];
   Double_t *p2 = &par[5];
   Double_t *p3 = &par[10];
   Double_t result = g2(x,p1) + g2(x,p2) + g2(x,p3);
   return result;
}

TCanvas *fit2a() {
   TCanvas *c = new TCanvas();
   gStyle->SetOptStat(kTRUE);
   gStyle->SetPalette(1);
   const Int_t npar = 15;
   Double_t f2params[npar] = {100,-3,3,-3,3,160,0,0.8,0,0.9,40,4,
0.7,4,0.7};
   TF2 *f2 = new TF2("f2",fun2,-10,10,-10,10, npar);
   f2->SetParameters(f2params);

   //Create an histogram and fill it randomly with f2
   TH2F *h2 = new TH2F("h2","From f2",40,-10,10,40,-10,10);
   Int_t nentries = 100000;
   h2->FillRandom("f2",nentries);
   //Fit h2 with original function f2
   Float_t ratio = 4*nentries/100000;
   f2params[ 0] *= ratio;
```

```
    f2params[ 5] *= ratio;
    f2params[10] *= ratio;
    f2->SetParameters(f2params);
    h2->Fit("f2","N");
    TCutG *cutg = new TCutG("cutg",5);//设置数据区域
    cutg->SetPoint(0,-7,-7);
    cutg->SetPoint(1, 2,-7);
    cutg->SetPoint(2, 2, 2);
    cutg->SetPoint(3,-7, 2);
    cutg->SetPoint(4,-7,-7);
    h2->Draw("lego2 0");
    h2->SetFillColor(38);
    f2->SetNpx(80);
    f2->SetNpy(80);
    f2->Draw("surf1 same bb [cutg]");
    return c;
}
```

```
// 图像上数据筛选
void fit2d()
{
   //example illustrating how to fit a 2-d histogram of type y=f
(x)
   // generate a 2-d histogram using a TCutG 图像上数据筛选
   const Int_t n = 6;
   Float_t x[n] = {0.092,0.83,0.94,0.81,0.12,0.1};
   Float_t y[n] = {0.71,9.4,9,8,0.3,0.71};
   TCutG *cut = new TCutG("cut",n,x,y);//设置数据区域
   TH2F *h2 = new TH2F("h2","h2",40,0,1,40,0,10);
   Float_t u,v;
   for (Int_t i=0;i<100000;i++) {
      u = gRandom->Rndm();
      v = 10*gRandom->Rndm();
      if (cut->IsInside(u,v)) h2->Fill(u,v);//筛选数据
   }
   TCanvas *c1 = new TCanvas("c1","show profile",600,900);
   c1->Divide(1,2);
   c1->cd(1);
   h2->Draw();
   c1->cd(2);

   //use a TProfile to convert the 2-d to 1-d problem
   TProfile *prof = h2->ProfileX();
   prof->Fit("pol1");
}
```

```
// make a contour plot and get the first contour in a TPolyMarker

void FirstContour()
{
   //this macro generates a color contour plot by selecting entr
ies from an ntuple file.
   //The TGraph object corresponding to the first contour line i
s accessed and displayed into a separate canvas.
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentM
acroName());
```

```
   dir.ReplaceAll("FirstContour.C","../hsimple.C");
   dir.ReplaceAll("/./","/");
   if (!gInterpreter->IsLoaded(dir.Data())) gInterpreter->LoadMa
cro(dir.Data());
   TFile *file = (TFile*)gROOT->ProcessLineFast("hsimple(1)");
   if (!file) return;
   TTree *ntuple = (TTree*)file->Get("ntuple");

   TCanvas *c1 = new TCanvas("c1","Contours",10,10,800,600);
   gStyle->SetPalette(1);
   ntuple->Draw("py:px","px*px+py*py < 20", "contz,list");

   //we must call Update to force the canvas to be painted.  Whe
n
   //painting the contour plot, the list of contours is generated

   //and a reference to it added to the Root list of special obj
ects
   c1->Update();

   TCanvas *c2 = new TCanvas("c2","First contour",100,100,800,600
);


   TObjArray *contours =
      (TObjArray*)gROOT->GetListOfSpecials()->FindObject("contou
rs");
   if (!contours) return;
   TList *lcontour1 = (TList*)contours->At(0);
   if (!lcontour1) return;
   TGraph *gc1 = (TGraph*)lcontour1->First();
   if (!gc1) return;
   if (gc1->GetN() < 10) return;
   gc1->SetMarkerStyle(21);
   gc1->Draw("alp");

   //We make a TCutG object with the array obtained from this gr
aph
   TCutG *cutg = new TCutG("cutg",gc1->GetN(),gc1->GetX(),gc1->G
etY());
```

```
   //We create a polymarker object with npmax points.
   const Int_t npmax = 50000;
   TPolyMarker *pm = new TPolyMarker(npmax);
   Int_t np = 0;
   while(1) {
      Double_t x = -4 +8*gRandom->Rndm();
      Double_t y = -4 +8*gRandom->Rndm();
      if (cutg->IsInside(x,y)) {
         pm->SetPoint(np,x,y);
         np++;
         if (np == npmax) break;
      }
   }
   pm->Draw();
}
```

TEllipse

TEventList

继承 TNamed

A list of selected entries in a TTree.

A TEventList object is a list of selected events (entries) in a TTree.

- Use function Enter to enter an element in the list
- The function Add may be used to merge two lists.
- The function Subtract may be used to subtract two lists.
- The function Reset may be used to reset a list.
- Use TEventList::Print(option) to print the contents. (option "all" prints all the list entries).
- Operators + and - correspond to functions Add and Subtract.
- A TEventList object can be saved on a file via the Write function.

class

```
   TEventList();/// Default constructor for a EventList.
   TEventList(const char *name, const char *title="",Int_t inits
ize=0, Int_t delta = 0);
/// Create a EventList.
/// This Eventlist is added to the list of objects in current di
rectory.

   TEventList(const TEventList &list);/// Copy constructor.
   virtual           ~TEventList();
   virtual void      Add(const TEventList *list);
/// Merge contents of alist with this list.
/// Both alist and this list are assumed to be sorted prior to t
his call

   virtual void      Clear(Option_t *option="") {Reset(option);}
   virtual Bool_t    Contains(Long64_t entry);/// Return TRUE if
 list contains entry.
   virtual Bool_t    ContainsRange(Long64_t entrymin, Long64_t e
```

```
ntrymax);
/// Return TRUE if list contains entries from entrymin to entrym
ax included.

   virtual void      DirectoryAutoAdd(TDirectory *);
/// Called by TKey and others to automatically add us to a direc
tory when we are read from a file.

   virtual void      Enter(Long64_t entry);/// Enter element ent
ry into the list.
   TDirectory       *GetDirectory() const {return fDirectory;}
   virtual Long64_t  GetEntry(Int_t index) const;
/// Return value of entry at index in the list.
/// Return -1 if index is not in the list range.

   virtual Int_t     GetIndex(Long64_t entry) const;
/// Return index in the list of element with value entry
/// array is supposed  to be sorted prior to this call.
/// If match is found, function returns position of element.
/// If no match found, function returns -1.

   virtual Long64_t *GetList() const { return fList; }
   virtual Int_t     GetN() const { return fN; }
   virtual Bool_t    GetReapplyCut() const { return fReapply; };
   virtual Int_t     GetSize() const { return fSize; }
   virtual void      Intersect(const TEventList *list);/// Remov
e elements from this list that are NOT present in alist.
   virtual Int_t     Merge(TCollection *list);/// Merge entries
in all the TEventList in the collection in this event list.
   virtual void      Print(Option_t *option="") const;/// Print
contents of this list.
   virtual void      Reset(Option_t *option="");/// Reset number
 of entries in event list.
   virtual void      Resize(Int_t delta=0);/// Resize list by de
lta entries.
   virtual void      SetDelta(Int_t delta=100) {fDelta = delta;}
   virtual void      SetDirectory(TDirectory *dir);
/// Remove reference to this EventList from current directory an
d add
/// reference to new directory dir. dir can be 0 in which case t
```

```
he list
/// does not belong to any directory.

   virtual void      SetName(const char *name); // *MENU* /// Ch
ange the name of this TEventList.
   virtual void      SetReapplyCut(Bool_t apply = kFALSE) {fReap
ply = apply;}; // *TOGGLE*
   virtual void      Sort();/// Sort list entries in increasing
order
   virtual void      Subtract(const TEventList *list);/// Remove
 elements from this list that are present in alist.

   TEventList&       operator=(const TEventList &list);

   friend TEventList operator+(const TEventList &list1, const TE
ventList &list2);
   friend TEventList operator-(const TEventList &list1, const TE
ventList &list2);
   friend TEventList operator*(const TEventList &list1, const TE
ventList &list2);
```

code

```
// A TEventList is automatically generated by TTree::Draw: examp
le

tree->Draw(">>elist1","x<0 && y> 0");

// In this example, a TEventList object named "elist1" will be
// generated. (Previous contents are overwritten).

tree->Draw(">>+elist1","x<0 && y> 0");

// In this example, selected entries are added to the list.

// The TEventList object is added to the list of objects in the
current directory.

// Use TTree:SetEventList(TEventList *list) to inform TTree that
 you
// want to use the list as input. The following code gets a poin
ter to
// the TEventList object created in the above commands:

TEventList *list = (TEventList*)gDirectory->Get("elist1");
```

example

TF1

继承于 TNamed, TAttLine, TAttFill, TAttMarker

A TF1 object is a 1-Dim function defined between a lower and upper limit.
The function may be a simple function based on a TFormula expression or a precompiled user function.
The function may have associated parameters.
TF1 graphics function is via the TH1 and TGraph drawing functions.

The following types of functions can be created:

1. Expression using variable x and no parameters
2. Expression using variable x with parameters
3. Lambda Expression with variable x and parameters
4. A general C function with parameters
5. A general C++ function object (functor) with parameters
6. A member function with parameters of a general C++ class

class

```
public:
    // TF1 status bits
    enum {
       kNotDraw     = BIT(9)  // don't draw the function when in
 a TH1
    };

   TF1();
   TF1(const char *name, const char *formula, Double_t xmin=0, D
ouble_t xmax = 1);
   TF1(const char *name, Double_t xmin, Double_t xmax, Int_t npa
r,Int_t ndim = 1);
#ifndef __CINT__
   TF1(const char *name, Double_t (*fcn)(Double_t *, Double_t *)
, Double_t xmin=0, Double_t xmax=1, Int_t npar=0, Int_t ndim = 1
```

```
); //fcn为自己定义的函数曲线，npar为调用的参数个数
   TF1(const char *name, Double_t (*fcn)(const Double_t *, const
 Double_t *), Double_t xmin=0, Double_t xmax=1, Int_t npar=0,Int
_t ndim = 1);
#endif

   // Constructors using functors (compiled mode only)
   TF1(const char *name, ROOT::Math::ParamFunctor f, Double_t xm
in = 0, Double_t xmax = 1, Int_t npar = 0,Int_t ndim = 1);

   // Template constructors from any  C++ callable object,  defi
ning  the operator() (double * , double *)
   // and returning a double.
   // The class name is not needed when using compile code, whil
e it is required when using
   // interpreted code via the specialized constructor with void
 *.
   // An instance of the C++ function class or its pointer can b
oth be used. The former is reccomended when using
   // C++ compiled code, but if CINT compatibility is needed, th
en a pointer to the function class must be used.
   // xmin and xmax specify the plotting range,  npar is the num
ber of parameters.
   // See the tutorial math/exampleFunctor.C for an example of u
sing this constructor
   template <typename Func>
   TF1(const char *name, Func f, Double_t xmin, Double_t xmax, I
nt_t npar,Int_t ndim = 1 );

   // backward compatible interface
   template <typename Func>
   TF1(const char *name, Func f, Double_t xmin, Double_t xmax, I
nt_t npar, const char *   ) :
      TNamed(name,name), TAttLine(), TAttFill(), TAttMarker(),
      fXmin(xmin), fXmax(xmax),
      fNpar(npar), fNdim(1),
      fNpx(100), fType(1),
      fNpfits(0), fNDF(0), fChisquare(0),
      fMinimum(-1111), fMaximum(-1111),
      fParErrors(std::vector<Double_t>(npar)),
```

```
      fParMin(std::vector<Double_t>(npar)),
      fParMax(std::vector<Double_t>(npar)),
      fParent(0), fHistogram(0),
      fMethodCall(0),
      fNormalized(false), fNormIntegral(0),
      fFunctor(ROOT::Math::ParamFunctor(f)),
      fFormula(0),
      fParams(new TF1Parameters(npar) )
   {
      DoInitialize();
   }


   // Template constructors from a pointer to any C++ class of t
ype PtrObj with a specific member function of type
   // MemFn.
   // The member function must have the signature of  (double *
, double *) and returning a double.
   // The class name and the method name are not needed when usi
ng compile code
   // (the member function pointer is used in this case), while
they are required when using interpreted
   // code via the specialized constructor with void *.
   // xmin and xmax specify the plotting range,  npar is the num
ber of parameters.
   // See the tutorial math/exampleFunctor.C for an example of u
sing this constructor
   template <class PtrObj, typename MemFn>
   TF1(const char *name, const  PtrObj& p, MemFn memFn, Double_t
 xmin, Double_t xmax, Int_t npar,Int_t ndim = 1) :
      TNamed(name,name), TAttLine(), TAttFill(), TAttMarker(),
      fXmin(xmin), fXmax(xmax),
      fNpar(npar), fNdim(ndim),
      fNpx(100), fType(1),
      fNpfits(0), fNDF(0), fChisquare(0),
      fMinimum(-1111), fMaximum(-1111),
      fParErrors(std::vector<Double_t>(npar)),
      fParMin(std::vector<Double_t>(npar)),
      fParMax(std::vector<Double_t>(npar)),
      fParent(0), fHistogram(0),
```

```
      fMethodCall(0),
      fNormalized(false), fNormIntegral(0),
      fFunctor   ( ROOT::Math::ParamFunctor(p,memFn) ),
      fFormula(0),
      fParams(new TF1Parameters(npar) )
   {
      DoInitialize();
   }
   // backward compatible interface
   template <class PtrObj, typename MemFn>
   TF1(const char *name, const  PtrObj& p, MemFn memFn, Double_t
 xmin, Double_t xmax, Int_t npar,const char * , const char * ) :
      TNamed(name,name), TAttLine(), TAttFill(), TAttMarker(),
      fXmin(xmin), fXmax(xmax),
      fNpar(npar), fNdim(1),
      fNpx(100), fType(1),
      fNpfits(0), fNDF(0), fChisquare(0),
      fMinimum(-1111), fMaximum(-1111),
      fParErrors(std::vector<Double_t>(npar)),
      fParMin(std::vector<Double_t>(npar)),
      fParMax(std::vector<Double_t>(npar)),
      fParent(0), fHistogram(0),
      fMethodCall(0),
      fNormalized(false), fNormIntegral(0),
      fFunctor   ( ROOT::Math::ParamFunctor(p,memFn) ),
      fFormula(0),
      fParams(new TF1Parameters(npar) )
   {
      DoInitialize();
   }

   TF1(const TF1 &f1);
   TF1& operator=(const TF1 &rhs);
   virtual   ~TF1();
   virtual void     AddParameter(const TString &name, Double_t v
alue) { if (fFormula) fFormula->AddParameter(name,value); }
   // virtual void     AddParameters(const pair<TString,Double_t
> *pairs, Int_t size) { fFormula->AddParameters(pairs,size); }
   // virtual void     AddVariable(const TString &name, Double_t
 value = 0) { if (fFormula) fFormula->AddVariable(name,value); }
```

```
   // virtual void     AddVariables(const TString *vars, Int_t s
ize) { if (fFormula) fFormula->AddVariables(vars,size); }
   virtual Bool_t   AddToGlobalList(Bool_t on = kTRUE);
/// Add to global list of functions (gROOT->GetListOfFunctions()
 )
/// return previous status (true of functions was already in the
 list false if not)

   virtual void     Browse(TBrowser *b);
   virtual void     Copy(TObject &f1) const;
   virtual Double_t Derivative (Double_t x, Double_t *params=0,
Double_t epsilon=0.001) const;
   virtual Double_t Derivative2(Double_t x, Double_t *params=0,
Double_t epsilon=0.001) const;
   virtual Double_t Derivative3(Double_t x, Double_t *params=0,
Double_t epsilon=0.001) const;
   static  Double_t DerivativeError();
   virtual Int_t    DistancetoPrimitive(Int_t px, Int_t py);
/// Compute distance from point px,py to a function.
/// Compute the closest distance of approach from point px,py to
 this
/// function. The distance is computed in pixels units.
/// Note that px is called with a negative value when the TF1 is
 in
/// TGraph or TH1 list of functions. In this case there is no po
int
/// looking at the histogram axis.

   virtual void     Draw(Option_t *option="");
/// Draw this function with its current attributes.
/// Possible option values are:
/// option | description
/// -------|----------------------------------------
/// "SAME" | superimpose on top of existing picture
/// "L"    | connect all computed points with a straight line
/// "C"    | connect all computed points with a smooth curve
/// "FC"   | draw a fill area below a smooth curve
/// Note that the default value is "L". Therefore to draw on top
/// of an existing picture, specify option "LSAME"
/// NB. You must use DrawCopy if you want to draw several times
```

```
the same
///     function in the current canvas.

   virtual TF1     *DrawCopy(Option_t *option="") const;
/// Draw a copy of this function with its current attributes.
///  This function MUST be used instead of Draw when you want to
 draw
///  the same function with different parameters settings in the
 same canvas.
/// Possible option values are:
/// option | description
/// -------|----------------------------------------
/// "SAME" | superimpose on top of existing picture
/// "L"    | connect all computed points with a straight line
/// "C"    | connect all computed points with a smooth curve
/// "FC"   | draw a fill area below a smooth curve
/// Note that the default value is "L". Therefore to draw on top
/// of an existing picture, specify option "LSAME"

   virtual TObject *DrawDerivative(Option_t *option="al"); // *M
ENU*
   virtual TObject *DrawIntegral(Option_t *option="al");   // *M
ENU*
   virtual void     DrawF1(Double_t xmin, Double_t xmax, Option_
t *option="");
   virtual Double_t Eval(Double_t x, Double_t y=0, Double_t z=0,
 Double_t t=0) const; //通过变量x获得其函数返回值
   virtual Double_t EvalPar(const Double_t *x, const Double_t *p
arams=0);
   virtual Double_t operator()(Double_t x, Double_t y=0, Double_
t z = 0, Double_t t = 0) const;
   virtual Double_t operator()(const Double_t *x, const Double_t
 *params=0);
   virtual void     ExecuteEvent(Int_t event, Int_t px, Int_t py)
;
   virtual void     FixParameter(Int_t ipar, Double_t value); //
/ Fix the value of a parameter. The specified value will be used
 in a fit operation
       Double_t     GetChisquare() const {return fChisquare;}
   virtual TH1     *GetHistogram() const; //将其转为TH1类型
```

```
   virtual TH1     *CreateHistogram() { return DoCreateHistogram
(fXmin, fXmax); }
   virtual TFormula *GetFormula() { return fFormula;}
   virtual const TFormula *GetFormula() const { return fFormula;
}
   virtual TString  GetExpFormula(Option_t *option="") const { r
eturn (fFormula) ? fFormula->GetExpFormula(option) : ""; }
   virtual const TObject *GetLinearPart(Int_t i) const { return
(fFormula) ? fFormula->GetLinearPart(i) : nullptr;}
   virtual Double_t GetMaximum(Double_t xmin=0, Double_t xmax=0,
 Double_t epsilon = 1.E-10, Int_t maxiter = 100, Bool_t logx = f
alse) const; /// Returns the maximum value of the function
///  First, the grid search is used to bracket the maximum
///  with the step size = (xmax-xmin)/fNpx.
///  This way, the step size can be controlled via the SetNpx()
function.
///  If the function is unimodal or if its extrema are far apart
, setting
///  the fNpx to a small value speeds the algorithm up many time
s.
///  Then, Brent's method is applied on the bracketed interval
///  epsilon (default = 1.E-10) controls the relative accuracy (
if |x| > 1 )
///  and absolute (if |x| < 1)  and maxiter (default = 100) cont
rols the maximum number
///  of iteration of the Brent algorithm
///  If the flag logx is set the grid search is done in log step
 size
///  This is done automatically if the log scale is set in the c
urrent Pad
/// NOTE: see also TF1::GetMaximumX and TF1::GetX

   virtual Double_t GetMinimum(Double_t xmin=0, Double_t xmax=0,
 Double_t epsilon = 1.E-10, Int_t maxiter = 100, Bool_t logx = f
alse) const; /// Returns the minimum value of the function on th
e (xmin, xmax) interval
///  First, the grid search is used to bracket the maximum
///  with the step size = (xmax-xmin)/fNpx. This way, the step s
ize
///  can be controlled via the SetNpx() function. If the functio
```

```
n is
///  unimodal or if its extrema are far apart, setting the fNpx
to
///  a small value speeds the algorithm up many times.
///  Then, Brent's method is applied on the bracketed interval
///  epsilon (default = 1.E-10) controls the relative accuracy (
if |x| > 1 )
///  and absolute (if |x| < 1)  and maxiter (default = 100) cont
rols the maximum number
///  of iteration of the Brent algorithm
///  If the flag logx is set the grid search is done in log step
 size
///  This is done automatically if the log scale is set in the c
urrent Pad
/// NOTE: see also TF1::GetMaximumX and TF1::GetX

   virtual Double_t GetMaximumX(Double_t xmin=0, Double_t xmax=0
, Double_t epsilon = 1.E-10, Int_t maxiter = 100, Bool_t logx =
false) const; /// Returns the X value corresponding to the maxim
um value of the function
   virtual Double_t GetMinimumX(Double_t xmin=0, Double_t xmax=0
, Double_t epsilon = 1.E-10, Int_t maxiter = 100, Bool_t logx =
false) const; /// Returns the X value corresponding to the minim
um value of the function
   virtual Double_t GetMaximumStored() const {return fMaximum;}
   virtual Double_t GetMinimumStored() const {return fMinimum;}
   virtual Int_t    GetNpar() const { return fNpar;}
   virtual Int_t    GetNdim() const { return fNdim;}
   virtual Int_t    GetNDF() const;
/// Return the number of degrees of freedom in the fit
/// the fNDF parameter has been previously computed during a fit.

/// The number of degrees of freedom corresponds to the number o
f points
/// used in the fit minus the number of free parameters.

   virtual Int_t    GetNpx() const {return fNpx;} //获得区间分成份数

    TMethodCall    *GetMethodCall() const {return fMethodCall;}
   virtual Int_t    GetNumber() const { return (fFormula) ? fFor
```

```
mula->GetNumber() : 0;}
   virtual Int_t    GetNumberFreeParameters() const;
   virtual Int_t    GetNumberFitPoints() const {return fNpfits;}
   virtual char    *GetObjectInfo(Int_t px, Int_t py) const;
        TObject    *GetParent() const {return fParent;}
   virtual Double_t GetParameter(Int_t ipar) const {
      return (fFormula) ? fFormula->GetParameter(ipar) : fParams
->GetParameter(ipar);
   }
   virtual Double_t GetParameter(const TString &name)  const {
      return (fFormula) ? fFormula->GetParameter(name) : fParams
->GetParameter(name);
   }
   virtual Double_t *GetParameters() const {
      return (fFormula) ? fFormula->GetParameters() : const_cast
<Double_t*>(fParams->GetParameters());
   }
   virtual void     GetParameters(Double_t *params) { if (fFormu
la) fFormula->GetParameters(params);
                                                        else std::
copy(fParams->ParamsVec().begin(), fParams->ParamsVec().end(), p
arams); }
   virtual const char *GetParName(Int_t ipar) const {
      return (fFormula) ? fFormula->GetParName(ipar) : fParams->
GetParName(ipar);
   }
   virtual Int_t    GetParNumber(const char* name) const {
      return (fFormula) ? fFormula->GetParNumber(name) : fParams
->GetParNumber(name);
   }
   virtual Double_t GetParError(Int_t ipar) const;
   virtual const Double_t *GetParErrors() const {return fParErro
rs.data();}
   virtual void     GetParLimits(Int_t ipar, Double_t &parmin, D
ouble_t &parmax) const;
   virtual Double_t GetProb() const;
   virtual Int_t    GetQuantiles(Int_t nprobSum, Double_t *q, co
nst Double_t *probSum);
   virtual Double_t GetRandom(); /// Return a random number foll
owing this function shape
```

```
/// The distribution contained in the function fname (TF1) is in
tegrated
/// over the channel contents.
/// It is normalized to 1.
/// For each bin the integral is approximated by a parabola.
/// The parabola coefficients are stored as non persistent data
members
/// Getting one random number implies:
///  - Generating a random number between 0 and 1 (say r1)
///  - Look in which bin in the normalized integral r1 correspon
ds to
///  - Evaluate the parabolic curve in the selected bin to find
the corresponding X value.
/// If the ratio fXmax/fXmin > fNpx the integral is tabulated in
 log scale in x
/// The parabolic approximation is very good as soon as the numb
er of bins is greater than 50.

   virtual Double_t GetRandom(Double_t xmin, Double_t xmax); ///
 Return a random number following this function shape in [xmin,x
max]
///   The distribution contained in the function fname (TF1) is
integrated
///   over the channel contents.
///   It is normalized to 1.
///   For each bin the integral is approximated by a parabola.
///   The parabola coefficients are stored as non persistent dat
a members
///   Getting one random number implies:
///     - Generating a random number between 0 and 1 (say r1)
///     - Look in which bin in the normalized integral r1 corres
ponds to
///     - Evaluate the parabolic curve in the selected bin to fi
nd
///       the corresponding X value.
///   The parabolic approximation is very good as soon as the nu
mber
///   of bins is greater than 50.
///  IMPORTANT NOTE
///  The integral of the function is computed at fNpx points. If
```

```
 the function
///  has sharp peaks, you should increase the number of points (
SetNpx)
///  such that the peak is correctly tabulated at several points.


   virtual void     GetRange(Double_t &xmin, Double_t &xmax) con
st;
   virtual void     GetRange(Double_t &xmin, Double_t &ymin, Dou
ble_t &xmax, Double_t &ymax) const;
   virtual void     GetRange(Double_t &xmin, Double_t &ymin, Dou
ble_t &zmin, Double_t &xmax, Double_t &ymax, Double_t &zmax) con
st;
   virtual Double_t GetSave(const Double_t *x);
   virtual Double_t GetX(Double_t y, Double_t xmin=0, Double_t x
max=0, Double_t epsilon = 1.E-10, Int_t maxiter = 100, Bool_t lo
gx = false) const;  //由y找x
/// Returns the X value corresponding to the function value fy f
or (xmin<x<xmax).
/// in other words it can find the roots of the function when fy
=0 and successive calls
/// by changing the next call to [xmin+eps,xmax] where xmin is t
he previous root.
///  First, the grid search is used to bracket the maximum
///  with the step size = (xmax-xmin)/fNpx. This way, the step s
ize
///  can be controlled via the SetNpx() function. If the functio
n is
///  unimodal or if its extrema are far apart, setting the fNpx
to
///  a small value speeds the algorithm up many times.
///  Then, Brent's method is applied on the bracketed interval
///  epsilon (default = 1.E-10) controls the relative accuracy (
if |x| > 1 )
///  and absolute (if |x| < 1)  and maxiter (default = 100) cont
rols the maximum number
///  of iteration of the Brent algorithm
///  If the flag logx is set the grid search is done in log step
 size
///  This is done automatically if the log scale is set in the c
```

```
urrent Pad
/// NOTE: see also TF1::GetMaximumX, TF1::GetMinimumX

   virtual Double_t GetXmin() const {return fXmin;}
   virtual Double_t GetXmax() const {return fXmax;}
   TAxis           *GetXaxis() const ;
   TAxis           *GetYaxis() const ;
   TAxis           *GetZaxis() const ;
   virtual Double_t GetVariable(const TString &name) { return (f
Formula) ? fFormula->GetVariable(name) : 0;}
   virtual Double_t GradientPar(Int_t ipar, const Double_t *x, D
ouble_t eps=0.01);
   virtual void     GradientPar(const Double_t *x, Double_t *gra
d, Double_t eps=0.01);
   virtual void     InitArgs(const Double_t *x, const Double_t *
params);
   static  void     InitStandardFunctions();
   virtual Double_t Integral(Double_t a, Double_t b, Double_t ep
srel=1.e-12);
   virtual Double_t IntegralOneDim(Double_t a, Double_t b, Doubl
e_t epsrel, Double_t epsabs, Double_t &err);
   virtual Double_t IntegralError(Double_t a, Double_t b, const
Double_t *params=0, const Double_t *covmat=0, Double_t epsilon=1
.E-2);
   virtual Double_t IntegralError(Int_t n, const Double_t * a, c
onst Double_t * b, const Double_t *params=0, const Double_t *cov
mat=0, Double_t epsilon=1.E-2);
   // virtual Double_t IntegralFast(const TGraph *g, Double_t a,
 Double_t b, Double_t *params=0);
   virtual Double_t IntegralFast(Int_t num, Double_t *x, Double_
t *w, Double_t a, Double_t b, Double_t *params=0, Double_t epsil
on=1e-12);
   virtual Double_t IntegralMultiple(Int_t n, const Double_t *a,
const Double_t *b, Int_t maxpts, Double_t epsrel, Double_t epsab
s ,Double_t &relerr,Int_t &nfnevl, Int_t &ifail);
   virtual Double_t IntegralMultiple(Int_t n, const Double_t *a,
const Double_t *b, Int_t /*minpts*/, Int_t maxpts, Double_t epsr
el, Double_t &relerr,Int_t &nfnevl, Int_t &ifail) {
      return  IntegralMultiple(n,a,b,maxpts, epsrel, epsrel, rel
err, nfnevl, ifail);
```

```
   }
   virtual Double_t IntegralMultiple(Int_t n, const Double_t *a,
const Double_t *b, Double_t epsrel, Double_t &relerr);
   virtual Bool_t   IsEvalNormalized() const { return fNormalize
d; }
   /// return kTRUE if the point is inside the function range
   virtual Bool_t   IsInside(const Double_t *x) const { return !
(  ( x[0] < fXmin) || ( x[0] > fXmax ) ); }
   virtual Bool_t   IsLinear() const { return (fFormula) ? fForm
ula->IsLinear() : false;}
   virtual Bool_t   IsValid() const;
   virtual void     Print(Option_t *option="") const;
   virtual void     Paint(Option_t *option="");
   virtual void     ReleaseParameter(Int_t ipar);
   virtual void     Save(Double_t xmin, Double_t xmax, Double_t
ymin, Double_t ymax, Double_t zmin, Double_t zmax);
   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
   virtual void     SetChisquare(Double_t chi2) {fChisquare = ch
i2;}
   virtual void     SetFitResult(const ROOT::Fit::FitResult & re
sult, const Int_t * indpar = 0);
   template <class PtrObj, typename MemFn>
   void SetFunction( PtrObj& p, MemFn memFn );
   template <typename Func>
   void SetFunction( Func f );
   virtual void     SetMaximum(Double_t maximum=-1111); // *MENU*

   virtual void     SetMinimum(Double_t minimum=-1111); // *MENU*

   virtual void     SetNDF(Int_t ndf);
   virtual void     SetNumberFitPoints(Int_t npfits) {fNpfits =
npfits;}
   virtual void     SetNormalized(Bool_t flag) { fNormalized = f
lag; Update(); }
   virtual void     SetNpx(Int_t npx=100); // *MENU* 如果取点太少
（也就是bin太大），图上显示的就是折线图连成的曲线，效果不好。为了曲线光滑，
需要将区间分成尽量多的份数。
   virtual void     SetParameter(Int_t param, Double_t value) {
      (fFormula) ? fFormula->SetParameter(param,value) : fParams
```

```
->SetParameter(param,value);
      Update();
   }
   virtual void     SetParameter(const TString &name, Double_t v
alue) {
      (fFormula) ? fFormula->SetParameter(name,value) : fParams-
>SetParameter(name,value);
      Update();
   }
   virtual void     SetParameters(const Double_t *params) {
      (fFormula) ? fFormula->SetParameters(params) : fParams->Se
tParameters(params);
      Update();
   }
   virtual void     SetParameters(Double_t p0,Double_t p1,Double
_t p2=0,Double_t p3=0,Double_t p4=0,
                                     Double_t p5=0,Double_t p6=0
,Double_t p7=0,Double_t p8=0,
                                     Double_t p9=0,Double_t p10=0
) {
      if (fFormula) fFormula->SetParameters(p0,p1,p2,p3,p4,p5,p6
,p7,p8,p9,p10);
      else          fParams->SetParameters(p0,p1,p2,p3,p4,p5,p6,
p7,p8,p9,p10);
      Update();
   } // *MENU*
   virtual void     SetParName(Int_t ipar, const char *name);
   virtual void     SetParNames(const char *name0="p0",const char
 *name1="p1",const char *name2="p2",
                                 const char *name3="p3",const char
 *name4="p4", const char *name5="p5",
                                 const char *name6="p6",const char
 *name7="p7",const char *name8="p8",
                                 const char *name9="p9",const char
 *name10="p10"); // *MENU*
   virtual void     SetParError(Int_t ipar, Double_t error);
   virtual void     SetParErrors(const Double_t *errors);
   virtual void     SetParLimits(Int_t ipar, Double_t parmin, Do
uble_t parmax);
   virtual void     SetParent(TObject *p=0) {fParent = p;}
```

```
   virtual void     SetRange(Double_t xmin, Double_t xmax); // *
MENU*
   virtual void     SetRange(Double_t xmin, Double_t ymin,  Doub
le_t xmax, Double_t ymax);
   virtual void     SetRange(Double_t xmin, Double_t ymin, Doubl
e_t zmin,  Double_t xmax, Double_t ymax, Double_t zmax);
   virtual void     SetSavedPoint(Int_t point, Double_t value);
   virtual void     SetTitle(const char *title=""); // *MENU*
   virtual void     Update();

   static  TF1     *GetCurrent();
   static  void     AbsValue(Bool_t reject=kTRUE);
/// Static function: set the fgAbsValue flag.
/// By default TF1::Integral uses the original function value to
 compute the integral
/// However, TF1::Moment, CentralMoment require to compute the i
ntegral
/// using the absolute value of the function.

   static  void     RejectPoint(Bool_t reject=kTRUE);
   static  Bool_t   RejectedPoint();
   static  void     SetCurrent(TF1 *f1);

   //Moments
   virtual Double_t Moment(Double_t n, Double_t a, Double_t b, c
onst Double_t *params=0, Double_t epsilon=0.000001);
   virtual Double_t CentralMoment(Double_t n, Double_t a, Double
_t b, const Double_t *params=0, Double_t epsilon=0.000001);
   virtual Double_t Mean(Double_t a, Double_t b, const Double_t
*params=0, Double_t epsilon=0.000001) {return Moment(1,a,b,param
s,epsilon);}
   virtual Double_t Variance(Double_t a, Double_t b, const Doubl
e_t *params=0, Double_t epsilon=0.000001) {return CentralMoment(2
,a,b,params,epsilon);}

   //some useful static utility functions to compute sampling po
ints for Integral
   //static  void     CalcGaussLegendreSamplingPoints(TGraph *g,
 Double_t eps=3.0e-11);
   //static  TGraph  *CalcGaussLegendreSamplingPoints(Int_t num=
```

```
21, Double_t eps=3.0e-11);
   static  void     CalcGaussLegendreSamplingPoints(Int_t num, D
ouble_t *x, Double_t *w, Double_t eps=3.0e-11);
```

一维直方图拟合参数选项：

```
TFitResultPtr TH1::Fit(const char *fname ,Option_t *option ,Opti
on_t *goption, Double_t xxmin, Double_t xxmax);
TFitResultPtr TH1::Fit(TF1 *f1 ,Option_t *option ,Option_t *gopt
ion, Double_t xxmin, Double_t xxmax);
//      The list of fit options is given in parameter option.
//         option = "W"  Set all weights to 1 for non empty bins
; ignore error bars
//                = "WW" Set all weights to 1 including empty bi
ns; ignore error bars
//                = "I"  Use integral of function in bin, normal
ized by the bin volume,
//                       instead of value at bin center
//                = "L"  Use Loglikelihood method (default is ch
isquare method)
//                = "WL" Use Loglikelihood method and bin conten
ts are not integer,
//                       i.e. histogram is weighted (must have S
umw2() set)
//                = "U"  Use a User specified fitting algorithm
(via SetFCN)
//                = "Q"  Quiet mode (minimum printing)
//                = "V"  Verbose mode (default is between Q and
V)
//                = "E"  Perform better Errors estimation using
Minos technique
//                = "B"  User defined parameter settings are use
d for predefined functions
//                       like "gaus", "expo", "poln", "landau".
//                       Use this option when you want to fix on
e or more parameters for these functions.
//                = "M"  More. Improve fit results.
//                       It uses the IMPROVE command of TMinuit
(see TMinuit::mnimpr).
```

```
//                        This algorithm attempts to improve the
found local minimum by searching for a
//                        better one.
//                = "R"  Use the Range specified in the function
 range
//                = "N"  Do not store the graphics function, do
not draw
//                = "0"  Do not plot the result of the fit. By d
efault the fitted function
//                        is drawn unless the option"N" above is
specified.
//                = "+"  Add this new fitted function to the lis
t of fitted functions
//                        (by default, any previous function is d
eleted)
//                = "C"  In case of linear fitting, don't calcul
ate the chisquare
//                        (saves time)
//                = "F"  If fitting a polN, switch to minuit fit
ter
//                = "S"  The result of the fit is returned in th
e TFitResultPtr
//                        (see below Access to the Fit Result)
//
//      When the fit is drawn (by default), the parameter goptio
n may be used
//      to specify a list of graphics options. See TH1::Draw for
 a complete
//      list of these options.
//
//      In order to use the Range option, one must first create
a function
//      with the expression to be fitted. For example, if your h
istogram
//      has a defined range between -4 and 4 and you want to fit
 a gaussian
//      only in the interval 1 to 3, you can do:
//           TF1 *f1 = new TF1("f1", "gaus", 1, 3);
//           histo->Fit("f1", "R");
//
```

```
//      Setting initial conditions
//      ==========================
//      Parameters must be initialized before invoking the Fit f
unction.
//      The setting of the parameter initial values is automatic
 for the
//      predefined functions : poln, expo, gaus, landau. One can
 however disable
//      this automatic computation by specifying the option "B".
//      Note that if a predefined function is defined with an ar
gument,
//      eg, gaus(0), expo(1), you must specify the initial value
s for
//      the parameters.
//      You can specify boundary limits for some or all paramete
rs via
//           f1->SetParLimits(p_number, parmin, parmax);
//      if parmin>=parmax, the parameter is fixed
//      Note that you are not forced to fix the limits for all p
arameters.
//      For example, if you fit a function with 6 parameters, yo
u can do:
//        func->SetParameters(0, 3.1, 1.e-6, -8, 0, 100);
//        func->SetParLimits(3, -10, -4);
//        func->FixParameter(4, 0);
//        func->SetParLimits(5, 1, 1);
//      With this setup, parameters 0->2 can vary freely
//      Parameter 3 has boundaries [-10,-4] with initial value -8

//      Parameter 4 is fixed to 0
//      Parameter 5 is fixed to 100.
//      When the lower limit and upper limit are equal, the para
meter is fixed.
//      However to fix a parameter to 0, one must call the FixPa
rameter function.
//
//      Note that option "I" gives better results but is slower.
//
//
//     Changing the fitting objective function
```

```
//     ========================================
//     By default a chi square function is used for fitting. Whe
n option "L" (or "LL") is used
//     a Poisson likelihood function (see note below) is used.
//     The functions are defined in the header Fit/Chi2Func.h or
 Fit/PoissonLikelihoodFCN and they
//     are implemented using the routines FitUtil::EvaluateChi2
or FitUtil::EvaluatePoissonLogL in
//     the file math/mathcore/src/FitUtil.cxx.
//     To specify a User defined fitting function, specify optio
n "U" and
//     call the following functions:
//       TVirtualFitter::Fitter(myhist)->SetFCN(MyFittingFunctio
n)
//     where MyFittingFunction is of type:
//     extern void MyFittingFunction(Int_t &npar, Double_t *gin,
 Double_t &f, Double_t *u, Int_t flag);
//
//     Chi2 Fits
//     =========
//     By default a chi2 (least-square) fit is performed on the
histogram. The so-called modified least-square method
//     is used where the residual for each bin is computed using
 as error the observed value (the bin error)
//
//     Chi2 = Sum{ ( y(i) - f (x(i) | p )/ e(i) )^2 }
//
//     where y(i) is the bin content for each bin i, x(i) is the
 bin center and e(i) is the bin error (sqrt(y(i) for
//     an un-weighted histogram. Bins with zero errors are exclu
ded from the fit. See also later the note on the treatment of em
pty bins.
//     When using option "I" the residual is computed not using
the function value at the bin center, f (x(i) | p), but the inte
gral
//     of the function in the bin,   Integral{ f(x|p)dx } divide
d by the bin volume
//
//     Likelihood Fits
//     ===============
```

```
//     When using option "L" a likelihood fit is used instead of
 the default chi2 square fit.
//     The likelihood is built assuming a Poisson probability de
nsity function for each bin.
//     The negative log-likelihood to be minimized is
//      NLL = Sum{ log Poisson( y(i) |{ f(x(i) | p ) ) }
//     The exact likelihood used is the Poisson likelihood descr
ibed in this paper:
//     S. Baker and R. D. Cousins, “Clarification of the use of
chi-square and likelihood functions in fits to histograms,”
//     Nucl. Instrum. Meth. 221 (1984) 437.
//     This method can then be used only when the bin content re
presents counts (i.e. errors are sqrt(N) ).
//     The likelihood method has the advantage of treating corre
ctly bins with low statistics. In case of high
//     statistics/bin the distribution of the bin content become
s a normal distribution and the likelihood and chi2 fit
//     give the same result.
//     The likelihood method, although a bit slower, it is there
fore the recommended method in case of low
//     bin statistics, where the chi2 method may give incorrect
results, in particular when there are
//     several empty bins (see also below).
//     In case of a weighted histogram, it is possible to perfor
m a likelihood fit by using the
//     option "WL". Note a weighted histogram is an histogram wh
ich has been filled with weights and it
//     contains the sum of the weight square ( TH1::Sumw2() has
been called). The bin error for a weighted
//     histogram is the square root of the sum of the weight squ
are.
//
//     Treatment of Empty Bins
//     ========================
//
//     Empty bins, which have the content equal to zero AND erro
r equal to zero,
//     are excluded by default from the chisquare fit, but they
are considered in the likelihood fit.
//     since they affect the likelihood if the function value in
```

```
 these bins is not negligible.
//     When using option "WW" these bins will be considered in t
he chi2 fit with an error of 1.
//     Note that if the histogram is having bins with zero conte
nt and non zero-errors they are considered as
//     any other bins in the fit. Instead bins with zero error a
nd non-zero content are excluded in the chi2 fit.
//     A likelihood fit should also not be peformed on such an h
istogram, since we are assuming a wrong pdf for each bin.
//     In general, one should not fit an histogram with non-empt
y bins and zero errors, apart if all the bins have zero errors.
//     In this case one could use the option "w", which gives a
weight=1 for each bin (unweighted least-square fit).
//
//     Fitting a histogram of dimension N with a function of dim
ension N-1
//     ==========================================================
==========
//     It is possible to fit a TH2 with a TF1 or a TH3 with a TF
2.
//     In this case the option "Integral" is not allowed and eac
h cell has
//     equal weight.
//
//     Associated functions
//     ====================
//     One or more object (typically a TF1*) can be added to the
 list
//     of functions (fFunctions) associated to each histogram.
//     When TH1::Fit is invoked, the fitted function is added to
 this list.
//     Given an histogram h, one can retrieve an associated func
tion
//     with:  TF1 *myfunc = h->GetFunction("myfunc");
//
//     Access to the fit result
//     ========================
//     The function returns a TFitResultPtr which can hold a  po
inter to a TFitResult object.
//     By default the TFitResultPtr contains only the status of
```

```
the fit which is return by an
//     automatic conversion of the TFitResultPtr to an integer.
One can write in this case directly:
//     Int_t fitStatus =  h->Fit(myFunc)
//
//     If the option "S" is instead used, TFitResultPtr contains
 the TFitResult and behaves as a smart
//     pointer to it. For example one can do:
//     TFitResultPtr r = h->Fit(myFunc,"S");
//     TMatrixDSym cov = r->GetCovarianceMatrix();  //  to acces
s the covariance matrix
//     Double_t chi2   = r->Chi2(); // to retrieve the fit chi2
//     Double_t par0   = r->Parameter(0); // retrieve the value
for the parameter 0
//     Double_t err0   = r->ParError(0); // retrieve the error f
or the parameter 0
//     r->Print("V");     // print full information of fit inclu
ding covariance matrix
//     r->Write();        // store the result in a file
//
//     The fit parameters, error and chi2 (but not covariance ma
trix) can be retrieved also
//     from the fitted function.
//     If the histogram is made persistent, the list of
//     associated functions is also persistent. Given a pointer
(see above)
//     to an associated function myfunc, one can retrieve the fu
nction/fit
//     parameters with calls such as:
//       Double_t chi2 = myfunc->GetChisquare();
//       Double_t par0 = myfunc->GetParameter(0); //value of 1st
 parameter
//       Double_t err0 = myfunc->GetParError(0);  //error on fir
st parameter
//
//     Access to the fit status
//     ========================
//     The status of the fit can be obtained converting the TFit
ResultPtr to an integer
//     independently if the fit option "S" is used or not:
```

```
//     TFitResultPtr r = h->Fit(myFunc,opt);
//     Int_t fitStatus = r;
//
//     The fitStatus is 0 if the fit is OK (i.e no error occurre
d).
//     The value of the fit status code is negative in case of a
n error not connected with the
//     minimization procedure, for example  when a wrong functio
n is used.
//     Otherwise the return value is the one returned from the m
inimization procedure.
//     When TMinuit (default case) or Minuit2 are used as minimi
zer the status returned is :
//     fitStatus =  migradResult + 10*minosResult + 100*hesseRes
ult + 1000*improveResult.
//     TMinuit will return 0 (for migrad, minos, hesse or improv
e) in case of success and 4 in
//     case of error (see the documentation of TMinuit::mnexcm).
 So for example, for an error
//     only in Minos but not in Migrad a fitStatus of 40 will be
 returned.
//     Minuit2 will return also 0 in case of success and differe
nt values in migrad minos or
//     hesse depending on the error. See in this case the docume
ntation of
//     Minuit2Minimizer::Minimize for the migradResult, Minuit2M
inimizer::GetMinosError for the
//     minosResult and Minuit2Minimizer::Hesse for the hesseResu
lt.
//     If other minimizers are used see their specific documenta
tion for the status code returned.
//     For example in the case of Fumili, for the status returne
d see TFumili::Minimize.
//
//     Excluding points
//     ================
//     Use TF1::RejectPoint inside your fitting function to excl
ude points
//     within a certain range from the fit. Example:
//     Double_t fline(Double_t *x, Double_t *par)
```

```
//     {
//         if (x[0] > 2.5 && x[0] < 3.5) {
//           TF1::RejectPoint();
//           return 0;
//        }
//        return par[0] + par[1]*x[0];
//     }
//
//     void exclude() {
//        TF1 *f1 = new TF1("f1", "[0] +[1]*x +gaus(2)", 0, 5);
//        f1->SetParameters(6, -1,5, 3, 0.2);
//        TH1F *h = new TH1F("h", "background + signal", 100, 0,
 5);
//        h->FillRandom("f1", 2000);
//        TF1 *fline = new TF1("fline", fline, 0, 5, 2);
//        fline->SetParameters(2, -1);
//        h->Fit("fline", "l");
//     }
//
//     Warning when using the option "0"
//     ==================================
//     When selecting the option "0", the fitted function is add
ed to
//     the list of functions of the histogram, but it is not dra
wn.
//     You can undo what you disabled in the following way:
//       h.Fit("myFunction", "0"); // fit, store function but do
 not draw
//       h.Draw(); function is not drawn
//       const Int_t kNotDraw = 1<<9;
//       h.GetFunction("myFunction")->ResetBit(kNotDraw);
//       h.Draw();  // function is visible again
//
//     Access to the Minimizer information during fitting
//     ==================================================
//     This function calls, the ROOT::Fit::FitObject function im
plemented in HFitImpl.cxx
//     which uses the ROOT::Fit::Fitter class. The Fitter class
creates the objective fuction
//     (e.g. chi2 or likelihood) and uses an implementation of t
```

```
he  Minimizer interface for minimizing
//     the function.
//     The default minimizer is Minuit (class TMinuitMinimizer w
hich calls TMinuit).
//     The default  can be set in the resource file in etc/syste
m.rootrc. For example
//     Root.Fitter:      Minuit2
//     A different fitter can also be set via ROOT::Math::Minimi
zerOptions::SetDefaultMinimizer
//     (or TVirtualFitter::SetDefaultFitter).
//     For example ROOT::Math::MinimizerOptions::SetDefaultMinim
izer("GSLMultiMin","BFGS");
//     will set the usdage of the BFGS algorithm of the GSL mult
i-dimensional minimization
//     (implemented in libMathMore). ROOT::Math::MinimizerOption
s can be used also to set other
//     default options, like maximum number of function calls, m
inimization tolerance or print
//     level. See the documentation of this class.
//
//     For fitting linear functions (containing the "++" sign" a
nd polN functions,
//     the linear fitter is automatically initialized.
```

code

```
TF1 *f1=new TF1("aaaaa","f(x)",min,max);
TF2 *f2=new TF2("aaaaa","f(x,y)",x min,x max,y min,y max);
```

```
// inline expression using standard C++ functions/operators
// inline expression using a ROOT function (e.g. from TMath) wit
hout parameters
TF1 *fa1 = new TF1("fa1","sin(x)/x",0,10);
fa1->Draw();

TF1 *fa2 = new TF1("fa2","TMath::DiLog(x)",0,10);
fa2->Draw();
```

```
// inline expression using a user defined CLING function by name
Double_t myFunc(x) { return x+sin(x); }
// ....
TF1 *fa3 = new TF1("fa3","myFunc(x)",-3,5);
fa3->Draw();
```

```
TF1 *fa = new TF1("fa","[0]*x*sin([1]*x)",-3,3);
fa->SetParameter(0,value_first_parameter);
fa->SetParameter(1,value_second_parameter);
fa->SetParName(0,"Constant");// Parameters may be given a name:
```

```
TCanvas *c = new TCanvas("c","c",0,0,500,300);
TF1 *fb2 = new TF1("fa3","TMath::Landau(x,[0],[1],0)",-5,10);
fb2->SetParameters(0.2,1.3); fb2->Draw();
```

```
// A lambda expression with variables and parameters **(NEW)**
// TF1 now supports using lambda expressions in the formula. Thi
s allows, by using a full C++ syntax the full power of lambda fu
nctions and still mantain the capability of storing the function
 in a file which cannot be done with funciton pointer or lambda
written not as expression, but as code (see items belows).
// Example on how using lambda to define a sum of two functions.
 Note that is necessary to provide the number of parameters
TF1 f1("f1","sin(x)",0,10);
TF1 f2("f2","cos(x)",0,10);
TF1 fsum("f1","[&](double *x, double *p){ return p[0]*f1(x) + p[
1]*f2(x); }",0,10,2);
```

example

```
   Double_t myfunction(Double_t *x, Double_t *par)
   {
      Float_t xx =x[0];
      Double_t f = TMath::Abs(par[0]*sin(par[1]*xx)/xx);
      return f;
   }
   void myfunc()
   {
      TF1 *f1 = new TF1("myfunc",myfunction,0,10,2);
      f1->SetParameters(2,1);
      f1->SetParNames("constant","coefficient");
      f1->Draw();
   }
   void myfit()
   {
      TH1F *h1=new TH1F("h1","test",100,0,10);
      h1->FillRandom("myfunc",20000);
      TF1 *f1=gROOT->GetFunction("myfunc");
      f1->SetParameters(800,1);
      h1->Fit("myfunc");
   }
```

```
   void gint() {
      TF1 *g = new TF1("g","gaus",-5,5);
      g->SetParameters(1,0,1);
      //default gaus integration method uses 6 points
      //not suitable to integrate on a large domain
      double r1 = g->Integral(0,5);
      double r2 = g->Integral(0,1000);

      //try with user directives computing more points
      Int_t np = 1000;
      double *x=new double[np];
      double *w=new double[np];
      g->CalcGaussLegendreSamplingPoints(np,x,w,1e-15);
      double r3 = g->IntegralFast(np,x,w,0,5);
      double r4 = g->IntegralFast(np,x,w,0,1000);
      double r5 = g->IntegralFast(np,x,w,0,10000);
      double r6 = g->IntegralFast(np,x,w,0,100000);
      printf("g->Integral(0,5)               = %g\n",r1);
      printf("g->Integral(0,1000)            = %g\n",r2);
      printf("g->IntegralFast(n,x,w,0,5)     = %g\n",r3);
      printf("g->IntegralFast(n,x,w,0,1000)  = %g\n",r4);
      printf("g->IntegralFast(n,x,w,0,10000) = %g\n",r5);
      printf("g->IntegralFast(n,x,w,0,100000)= %g\n",r6);
      delete [] x;
      delete [] w;
   }
///   This example produces the following results:
///
///      g->Integral(0,5)               = 1.25331
///      g->Integral(0,1000)            = 1.25319
///      g->IntegralFast(n,x,w,0,5)     = 1.25331
///      g->IntegralFast(n,x,w,0,1000)  = 1.25331
///      g->IntegralFast(n,x,w,0,10000) = 1.25331
///      g->IntegralFast(n,x,w,0,100000)= 1.253
```

TF2

class

code

example

```
//由 TF2 函数填充 TH2
gROOT->Reset();
gStyle->SetOptStat(0);
gStyle->SetPalette(1);
gStyle->SetCanvasColor(33);
gStyle->SetFrameFillColor(18);
TF2 *f2 = new TF2("f2","xygaus + xygaus(5) + xylandau(10)",-4,4,
-4,4);
Double_t params[] = {130,-1.4,1.8,1.5,1, 150,2,0.5,-2,0.5, 3600,
-2,0.7,-3,0.3};
f2->SetParameters(params);
TH2F h2("h2","xygaus + xygaus(5) + xylandau(10)",20,-4,4,20,-4,4)
;
h2.SetFillColor(46);
h2.FillRandom("f2",40000);
```

TF3

TFile

继承TDirectoryFile

A ROOT file is a suite of consecutive data records (TKey instances) with a well defined format.

If the key is located past the 32 bit file limit (> 2 GB) then some fields will be 8 instead of 4 bytes:

```
Byte Range      | Member Name | Description
----------------|-----------|--------------
1->4            | Nbytes    | Length of compressed object (in by
tes)
5->6            | Version   | TKey version identifier
7->10           | ObjLen    | Length of uncompressed object
11->14          | Datime    | Date and time when object was writ
ten to file
15->16          | KeyLen    | Length of the key structure (in by
tes)
17->18          | Cycle     | Cycle of key
19->22 [19->26] | SeekKey   | Pointer to record itself (consiste
ncy check)
23->26 [27->34] | SeekPdir  | Pointer to directory header
27->27 [35->35] | lname     | Number of bytes in the class name
28->.. [36->..] | ClassName | Object Class Name
..->..          | lname     | Number of bytes in the object name
..->..          | Name      | lName bytes with the name of the o
bject
..->..          | lTitle    | Number of bytes in the object titl
e
..->..          | Title     | Title of the object
----->          | DATA      | Data bytes associated to the objec
t
```

The first data record starts at byte fBEGIN (currently set to kBEGIN). Bytes 1->kBEGIN contain the file description, when fVersion >= 1000000 it is a large file (> 2 GB) and the offsets will be 8 bytes long and fUnits will be set to 8:

```
Byte Range      | Record Name | Description
----------------|-------------|------------
1->4            | "root"      | Root file identifier
5->8            | fVersion    | File format version
9->12           | fBEGIN      | Pointer to first data record
13->16 [13->20] | fEND        | Pointer to first free word at th
e EOF
17->20 [21->28] | fSeekFree   | Pointer to FREE data record
21->24 [29->32] | fNbytesFree | Number of bytes in FREE data rec
ord
25->28 [33->36] | nfree       | Number of free data records
29->32 [37->40] | fNbytesName | Number of bytes in TNamed at cre
ation time
33->33 [41->41] | fUnits      | Number of bytes for file pointer
s
34->37 [42->45] | fCompress   | Compression level and algorithm
38->41 [46->53] | fSeekInfo   | Pointer to TStreamerInfo record
42->45 [54->57] | fNbytesInfo | Number of bytes in TStreamerInfo
 record
46->63 [58->75] | fUUID       | Universal Unique ID
```

```
/// A ROOT file is designed such that one can write in the file
in pure
/// sequential mode (case of BATCH jobs). In this case, the file
 may be
/// read sequentially again without using the file index written
/// at the end of the file. In case of a job crash, all the info
rmation
/// on the file is therefore protected.
/// A ROOT file can be used interactively. In this case, one has
 the
/// possibility to delete existing objects and add new ones.
/// When an object is deleted from the file, the freed space is
added
```

```
/// into the FREE linked list (fFree). The FREE list consists of
 a chain
/// of consecutive free segments on the file. At the same time,
the first
/// 4 bytes of the freed record on the file are overwritten by G
APSIZE
/// where GAPSIZE = -(Number of bytes occupied by the record).
/// Option compress is used to specify the compression level and
 algorithm:
///
///     compress = 100 * algorithm + level
///
/// Level | Explanation
/// ------|-------------
/// 0   | objects written to this file will not be compressed.
/// 1   | minimal compression level but fast.
/// ... | ....
/// 9   | maximal compression level but slower and might use mor
e memory.
/// (For the currently supported algorithms, the maximum level i
s 9)
/// If compress is negative it indicates the compression level i
s not set yet.
/// The enumeration ROOT::ECompressionAlgorithm associates each
/// algorithm with a number. There is a utility function to help
/// to set the value of compress. For example,
///     ROOT::CompressionSettings(ROOT::kLZMA, 1)
/// will build an integer which will set the compression to use
/// the LZMA algorithm and compression level 1.  These are defin
ed
/// in the header file <em>Compression.h</em>.
/// Note that the compression settings may be changed at any tim
e.
/// The new compression settings will only apply to branches cre
ated
/// or attached after the setting is changed and other objects w
ritten
/// after the setting is changed.
```

class

```
   /// TFile status bits. BIT(13) is taken up by TObject
   enum EStatusBits {
      kRecovered     = BIT(10),
      kHasReferences = BIT(11),
      kDevNull       = BIT(12),
      kWriteError    = BIT(14),
      kBinaryFile    = BIT(15),
      kRedirected    = BIT(16)
   };
   enum ERelativeTo { kBeg = 0, kCur = 1, kEnd = 2 };
   enum { kStartBigFile  = 2000000000 };
   /// File type
   enum EFileType { kDefault = 0, kLocal = 1, kNet = 2, kWeb = 3
, kFile = 4, kMerge = 5};

   TFile();

   /// \param[in] fname1 The name of the file
   /// \param[in] option Specifies the mode in which the file is
 opened
   /// \param[in] ftitle The title of the file
   /// \param[in] compress Specifies the compression algorithm a
nd level
   ///
   /// will be used by object browsers to automatically identify
 the file as
   /// a ROOT file. If the constructor fails in any way IsZombie
() will
   /// return true. Use IsOpen() to check if the file is (still)
 open.
   /// To open non-local files use the static TFile::Open() meth
od, that
   /// will take care of opening the files using the correct rem
ote file
   /// access plugin.
   ///
   /// Option | Description
```

```
   /// -------|------------
   /// NEW or CREATE | Create a new file and open it for writing
, if the file already exists the file is not opened.
   /// RECREATE      | Create a new file, if the file already ex
ists it will be overwritten.
   /// UPDATE        | Open an existing file for writing. If no
file exists, it is created.
   /// READ          | Open an existing file for reading (defaul
t).
   /// NET           | Used by derived remote file access classe
s, not a user callable option.
   /// WEB           | Used by derived remote http access class,
 not a user callable option.
   TFile(const char *fname, Option_t *option="", const char *fti
tle="", Int_t compress=1);


   virtual ~TFile();
   virtual void        Close(Option_t *option=""); // *MENU*
/// Close a file.
/// \param[in] option If option == "R", all TProcessIDs referenc
ed by this file are deleted.
/// Calling TFile::Close("R") might be necessary in case one rea
ds a long list
/// of files having TRef, writing some of the referenced objects
 or TRef
/// to a new file. If the TRef or referenced objects of the file
 being closed
/// will not be referenced again, it is possible to minimize the
 size
/// of the TProcessID data structures in memory by forcing a del
ete of
/// the unused TProcessID.

   virtual void        Copy(TObject &) const { MayNotUse("Copy(T
Object &)"); }
   virtual Bool_t      Cp(const char *dst, Bool_t progressbar =
kTRUE,UInt_t buffersize = 1000000);
/// Allows to copy this file to the dst URL. Returns kTRUE in ca
se of success,
```

```
/// kFALSE otherwise.

   virtual TKey*       CreateKey(TDirectory* mother, const TObje
ct* obj, const char* name, Int_t bufsize);
/// Creates key for object and converts data to buffer.

   virtual TKey*       CreateKey(TDirectory* mother, const void*
 obj, const TClass* cl,
                                 const char* name, Int_t bufsize)
;
/// Creates key for object and converts data to buffer.

   static TFile      *&CurrentFile(); // Return the current file
 for this thread.
/// Return the current ROOT file if any.
/// Note that if 'cd' has been called on a TDirectory that does
not belong to a file,
/// gFile will be unchanged and still points to the file of the
previous current
/// directory that was a file.

   virtual void        Delete(const char *namecycle="");
/// Delete object namecycle.
/// \param[in] namecycle Encodes the name and cycle of the objec
ts to delete
/// Namecycle identifies an object in the top directory of the f
ile namecycle
/// has the format <em>name;cycle</em>.
///   - <em>name  = *</em> means all objects
///   - <em>cycle = *</em> means all cycles (memory and keys)
///   - <em>cycle = ""</em> or cycle = 9999 ==> apply to a memor
y object
/// When name=* use T* to delete subdirectories also
///
/// Examples:
/// name/cycle | Action
/// -----------|-------
/// foo   | delete object named foo in memory
/// foo;1 | delete cycle 1 of foo on file
/// foo;* | delete all cycles of foo on disk and also from memory
```

```
/// *;2   | delete all objects on file having the cycle 2
/// *;*   | delete all objects from memory and file
/// T*;*  | delete all objects from memory and file and all subd
irectories

   virtual void        Draw(Option_t *option="");
/// Fill Graphics Structure and Paint.
/// Loop on all objects (memory or file) and all subdirectories.

   virtual void        DrawMap(const char *keys="*",Option_t *op
tion=""); // *MENU*
/// Draw map of objects in this file.

   virtual void        FillBuffer(char *&buffer);
/// Encode file output buffer.
/// The file output buffer contains only the FREE data record.

   virtual void        Flush();
/// Synchronize a file's in-memory and on-disk states.

   TArchiveFile       *GetArchive() const { return fArchive; }
   Long64_t            GetArchiveOffset() const { return fArchiv
eOffset; }
   Int_t               GetBestBuffer() const;
/// Return the best buffer size of objects on this file.
/// The best buffer size is estimated based on the current mean
value
/// and standard deviation of all objects written so far to this
 file.
/// Returns mean value + one standard deviation.

   virtual Int_t       GetBytesToPrefetch() const;
/// Max number of bytes to prefetch.
/// By default this is 75% of the
/// read cache size. But specific TFile implementations may need
 to change it

   TFileCacheRead     *GetCacheRead(TObject* tree = 0) const;
   TFileCacheWrite    *GetCacheWrite() const;
```

```
   TArrayC            *GetClassIndex() const { return fClassInde
x; }
   Int_t               GetCompressionAlgorithm() const;
   Int_t               GetCompressionLevel() const;
   Int_t               GetCompressionSettings() const;
   Float_t             GetCompressionFactor();
/// Return the file compression factor.
/// Add total number of compressed/uncompressed bytes for each k
ey.
/// Returns the ratio of the two.

   virtual Long64_t    GetEND() const { return fEND; }
   virtual Int_t       GetErrno() const;
   virtual void        ResetErrno() const;
   Int_t               GetFd() const { return fD; }
   virtual const TUrl *GetEndpointUrl() const { return &fUrl; }
   TObjArray          *GetListOfProcessIDs() const {return fProc
essIDs;}
   TList              *GetListOfFree() const { return fFree; }
   virtual Int_t       GetNfree() const { return fFree->GetSize(
); }
   virtual Int_t       GetNProcessIDs() const { return fNProcess
IDs; }
   Option_t           *GetOption() const { return fOption.Data()
; }
   virtual Long64_t    GetBytesRead() const { return fBytesRead;
 }
   virtual Long64_t    GetBytesReadExtra() const { return fBytes
ReadExtra; }
   virtual Long64_t    GetBytesWritten() const;
/// Return the total number of bytes written so far to the file.

   virtual Int_t       GetReadCalls() const { return fReadCalls;
 }
   Int_t               GetVersion() const { return fVersion; }
   Int_t               GetRecordHeader(char *buf, Long64_t first
, Int_t maxbytes,
                                       Int_t &nbytes, Int_t &obj
len, Int_t &keylen);
/// Read the logical record header starting at a certain postion.
```

```
/// \param[in] maxbytes Bytes which are read into buf.
/// \param[out] nbytes Number of bytes in record if negative, th
is is a deleted
/// record if 0, cannot read record, wrong value of argument fir
st
/// \param[out] objlen Uncompressed object size
/// \param[out] keylen Length of logical record header
/// The function reads nread bytes
/// where nread is the minimum of maxbytes and the number of byt
es
/// before the end of file. The function returns nread.
/// Note that the arguments objlen and keylen are returned only
/// if maxbytes >=16

   virtual Int_t       GetNbytesInfo() const {return fNbytesInfo
;}
   virtual Int_t       GetNbytesFree() const {return fNbytesFree
;}
   virtual TString     GetNewUrl() { return ""; }
   Long64_t            GetRelOffset() const { return fOffset - f
ArchiveOffset; }
   virtual Long64_t    GetSeekFree() const {return fSeekFree;}
   virtual Long64_t    GetSeekInfo() const {return fSeekInfo;}
   virtual Long64_t    GetSize() const;
/// Returns the current file size. Returns -1 in case the file c
ould not
/// be stat'ed.

   virtual TList      *GetStreamerInfoList();
/// Read the list of TStreamerInfo objects written to this file.
/// The function returns a TList. It is the user's responsibility

/// to delete the list created by this function.

   const   TList      *GetStreamerInfoCache();
   virtual void        IncrementProcessIDs() { fNProcessIDs++; }
   virtual Bool_t      IsArchive() const { return fIsArchive; }
           Bool_t      IsBinary() const { return TestBit(kBinary
File); }
```

```
            Bool_t      IsRaw() const { return !fIsRootFile; }
   virtual Bool_t       IsOpen() const;
   virtual void         ls(Option_t *option="") const;
/// List file contents.
/// Indentation is used to identify the file tree.
/// Subdirectories are listed first, then objects in memory,
/// then objects on the file.

   virtual void         MakeFree(Long64_t first, Long64_t last);
/// Mark unused bytes on the file.
/// The list of free segments is in the fFree linked list.
/// When an object is deleted from the file, the freed space is
added
/// into the FREE linked list (fFree). The FREE list consists of
 a chain
/// of consecutive free segments on the file. At the same time,
the first
/// 4 bytes of the freed record on the file are overwritten by G
APSIZE
/// where GAPSIZE = -(Number of bytes occupied by the record).

   virtual void         MakeProject(const char *dirname, const ch
ar *classes="*",
                                    Option_t *option="new"); // *
MENU*
/// Generate source code necessary to access the objects stored
in the file.
/// Generate code in directory dirname for all classes specified
 in
/// argument classes If classes = "*" (default and currently the
/// only supported value), the function generates an include file

/// for each class in the StreamerInfo list for which a TClass
/// object does not exist.
///
/// The code generated includes:
///   - dirnameProjectHeaders.h, which contains one #include sta
tement per generated header file
///   - dirnameProjectSource.cxx,which contains all the construc
tors and destructors implementation.
```

```
/// and one header per class that is not nested inside another c
lass.
/// The header file name is the fully qualified name of the clas
s after all the special characters
/// "<>,:" are replaced by underscored.  For example for std::pa
ir<edm::Vertex,int> the file name is
/// pair_edm__Vertex_int_.h
///
/// In the generated classes, map, multimap when the first templ
ate parameter is a class
/// are replaced by a vector of pair. set and multiset when the
tempalte parameter
/// is a class are replaced by a vector. This is required since
we do not have the
/// code needed to order and/or compare the object of the classe
s.
/// This is a quick explanation of the options available:
/// Option | Details
/// -------|--------
/// new (default) | A new directory dirname is created. If dirna
me already exist, an error message is printed and the function r
eturns.
/// recreate      | If dirname does not exist, it is created (li
ke in "new"). If dirname already exist, all existing files in di
rname are deleted before creating the new files.
/// update        | New classes are added to the existing direct
ory. Existing classes with the same name are replaced by the new
 definition. If the directory dirname doest not exist, same effe
ct as "new".
/// genreflex     | Use genreflex rather than rootcint to genera
te the dictionary.
/// par           | Create a PAR file with the minimal set of co
de needed to read the content of the ROOT file. The name of the
PAR file is basename(dirname), with extension '.par' enforced; t
he PAR file will be created at dirname(dirname).
///
/// If, in addition to one of the 3 above options, the option "+
" is specified,
/// the function will generate:
///   - a script called MAKEP to build the shared lib
```

```
///   - a dirnameLinkDef.h file
///   - rootcint will be run to generate a dirnameProjectDict.cx
x file
///   - dirnameProjectDict.cxx will be compiled with the current
 options in compiledata.h
///   - a shared lib dirname.so will be created.
/// If the option "++" is specified, the generated shared lib is
 dynamically
/// linked with the current executable module.
/// If the option "+" and "nocompile" are specified, the utility
 files are generated
/// as in the option "+" but they are not executed.
/// Example:
///  file.MakeProject("demo","*","recreate++");
///   - creates a new directory demo unless it already exist
///   - clear the previous directory content
///   - generate the xxx.h files for all classes xxx found in th
is file
///    and not yet known to the CINT dictionary.
///   - creates the build script MAKEP
///   - creates a LinkDef.h file
///   - runs rootcint generating demoProjectDict.cxx
///   - compiles demoProjectDict.cxx into demoProjectDict.o
///   - generates a shared lib demo.so
///   - dynamically links the shared lib demo.so to the executab
le
///  If only the option "+" had been specified, one can still li
nk the
///  shared lib to the current executable module with:
///      gSystem->load("demo/demo.so");
/// The following feature is not yet enabled:
/// One can restrict the list of classes to be generated by usin
g expressions like:
///
///     classes = "Ali*" generate code only for classes starting
 with Ali
///     classes = "myClass" generate code for class MyClass only.


   virtual void        Map(); // *MENU*
```

```
   virtual Bool_t      Matches(const char *name);
/// Return kTRUE if 'url' matches the coordinates of this file.
/// The check is implementation dependent and may need to be ove
rload
/// by each TFile implementation relying on this check.
/// The default implementation checks the file name only.

   virtual Bool_t      MustFlush() const {return fMustFlush;}
   virtual void        Paint(Option_t *option="");
/// Paint all objects in the file.

   virtual void        Print(Option_t *option="") const;
/// Print all objects in the file.

   virtual Bool_t      ReadBufferAsync(Long64_t offs, Int_t len)
;
   virtual Bool_t      ReadBuffer(char *buf, Int_t len);
/// Read a buffer from the file. This is the basic low level rea
d operation.
/// Returns kTRUE in case of failure.

   virtual Bool_t      ReadBuffer(char *buf, Long64_t pos, Int_t
 len);
/// Read a buffer from the file at the offset 'pos' in the file.
/// Returns kTRUE in case of failure.
/// Compared to ReadBuffer(char*, Int_t), this routine does _not_

/// change the cursor on the physical file representation (fD)
/// if the data is in this TFile's cache.

   virtual Bool_t      ReadBuffers(char *buf, Long64_t *pos, Int
_t *len, Int_t nbuf);
/// Read the nbuf blocks described in arrays pos and len.
/// The value pos[i] is the seek position of block i of length l
en[i].
/// Note that for nbuf=1, this call is equivalent to TFile::Reaf
Buffer.
/// This function is overloaded by TNetFile, TWebFile, etc.
/// Returns kTRUE in case of failure.
```

```
   virtual void        ReadFree();
/// Read the FREE linked list.
/// Every file has a linked list (fFree) of free segments.
/// This linked list has been written on the file via WriteFree
/// as a single data record.

   virtual TProcessID *ReadProcessID(UShort_t pidf);
/// The TProcessID with number pidf is read from this file.
/// If the object is not already entered in the gROOT list, it i
s added.

   virtual void        ReadStreamerInfo();
/// Read the list of StreamerInfo from this file.
/// The key with name holding the list of TStreamerInfo objects
is read.
/// The corresponding TClass objects are updated.
/// Note that this function is not called if the static member f
gReadInfo is false.
/// (see TFile::SetReadStreamerInfo)

   virtual Int_t       Recover();
/// Attempt to recover file if not correctly closed
/// The function returns the number of keys that have been recov
ered.
/// If no keys can be recovered, the file will be declared Zombi
e by
/// the calling function. This function is automatically called
when
/// opening a file.
/// If the file is open in read only mode, the file is not modif
ied.
/// If open in update mode and the function finds something to r
ecover,
/// a new directory header is written to the file. When opening
the file gain
/// no message from Recover will be reported.
/// If keys have been recovered, the file is usable and you can
safely
/// read the corresponding objects.
/// If the file is not usable (a zombie), you can test for this
```

```
case
/// with code like:
/// ~~~{.cpp}
/// TFile f("myfile.root");
/// if (f.IsZombie()) {<actions to take if file is unusable>}
/// ~~~
/// If the file has been recovered, the bit kRecovered is set in
 the TFile object in memory.
/// You can test if the file has been recovered with
///     if (f.TestBit(TFile::kRecovered)) {... the file has been
 recovered}
/// When writing TTrees to a file, it is important to save the T
ree header
/// at regular intervals (see TTree::AutoSave). If a file contai
ning a Tree
/// is recovered, the last Tree header written to the file will
be used.
/// In this case all the entries in all the branches written bef
ore writing
/// the header are valid entries.
/// One can disable the automatic recovery procedure by setting
///     TFile.Recover 0
/// in the system.rootrc file.

   virtual Int_t       ReOpen(Option_t *mode);
/// Reopen a file with a different access mode.
/// For example, it is possible to change from READ to
/// UPDATE or from NEW, CREATE, RECREATE, UPDATE to READ. Thus t
he
/// mode argument can be either "READ" or "UPDATE". The method r
eturns
/// 0 in case the mode was successfully modified, 1 in case the
mode
/// did not change (was already as requested or wrong input argu
ments)
/// and -1 in case of failure, in which case the file cannot be
used
/// anymore. The current directory (gFile) is changed to this fi
le.
```

```
   virtual void        Seek(Long64_t offset, ERelativeTo pos = k
Beg);
/// Seek to a specific position in the file. Pos it either kBeg,
 kCur or kEnd.

   virtual void        SetCacheRead(TFileCacheRead *cache, TObje
ct* tree = 0, ECacheAction action = kDisconnect);
/// Set a pointer to the read cache.
/// This relinquishes ownership</b> of the previous cache, so if
 you do not
/// already have a pointer to the previous cache (and there was
a previous
/// cache), you ought to retrieve (and delete it if needed) usin
g:
///     TFileCacheRead *older = myfile->GetCacheRead();
/// The action specifies how to behave when detaching a cache fr
om the
/// the TFile.  If set to (default) kDisconnect, the contents of
 the cache
/// will be flushed when it is removed from the file, and it wil
l disconnect
/// the cache object from the file.  In almost all cases, this i
s what you want.
/// If you want to disconnect the cache temporarily from this tr
ee and re-attach
/// later to the same fil, you can set action to kDoNotDisconnec
t.  This will allow
/// things like prefetching to continue in the background while
it is no longer the
/// default cache for the TTree.  Except for a few expert use ca
ses, kDisconnect is
/// likely the correct setting.
/// WARNING: if action=kDoNotDisconnect, you MUST delete the cac
he before TFile.

   virtual void        SetCacheWrite(TFileCacheWrite *cache);
/// Set a pointer to the write cache.
/// If file is null the existing write cache is deleted.

   virtual void        SetCompressionAlgorithm(Int_t algorithm=0)
```

```
;
/// See comments for function SetCompressionSettings

   virtual void        SetCompressionLevel(Int_t level=1);
/// See comments for function SetCompressionSettings

   virtual void        SetCompressionSettings(Int_t settings=1);
/// Used to specify the compression level and algorithm.
/// See the TFile constructor for the details.

   virtual void        SetEND(Long64_t last) { fEND = last; }
   virtual void        SetOffset(Long64_t offset, ERelativeTo po
s = kBeg);
/// Set position from where to start reading.

   virtual void        SetOption(Option_t *option=">") { fOption
 = option; }
   virtual void        SetReadCalls(Int_t readcalls = 0) { fRead
Calls = readcalls; }
   virtual void        ShowStreamerInfo();
/// Show the StreamerInfo of all classes written to this file.

   virtual Int_t       Sizeof() const;
   void                SumBuffer(Int_t bufsize);
   virtual Bool_t      WriteBuffer(const char *buf, Int_t len);
/// Write a buffer to the file. This is the basic low level writ
e operation.
/// Returns kTRUE in case of failure.

   virtual Int_t       Write(const char *name=0, Int_t opt=0, In
t_t bufsiz=0);
/// Write memory objects to this file.
/// Loop on all objects in memory (including subdirectories).
/// A new key is created in the KEYS linked list for each object.

/// The list of keys is then saved on the file (via WriteKeys)
/// as a single data record.
/// For values of opt see TObject::Write().
/// The directory header info is rewritten on the directory head
er record.
```

```
/// The linked list of FREE segments is written.
/// The file header is written (bytes 1->fBEGIN).

   virtual Int_t       Write(const char *name=0, Int_t opt=0, In
t_t bufsiz=0) const;
   virtual void        WriteFree();
/// Write FREE linked list on the file.
/// The linked list of FREE segments (fFree) is written as a sin
gle data
/// record.

   virtual void        WriteHeader();
/// Write File Header.

   virtual UShort_t    WriteProcessID(TProcessID *pid);
/// Check if the ProcessID pidd is already in the file,
/// if not, add it and return the index  number in the local fil
e list.

   virtual void        WriteStreamerInfo();
/// Write the list of TStreamerInfo as a single object in this f
ile
/// The class Streamer description for all classes written to th
is file
/// is saved. See class TStreamerInfo.

   static TFileOpenHandle
                      *AsyncOpen(const char *name, Option_t *opt
ion = "",
                                 const char *ftitle = "", Int_t
compress = 1,
                                 Int_t netopt = 0);
/// Submit an asynchronous open request.
/// See TFile::Open(const char *, ...) for an
/// explanation of the arguments. A handler is returned which is
 to be passed
/// to TFile::Open(TFileOpenHandle *) to get the real TFile inst
ance once
/// the file is open.
/// This call never blocks and it is provided to allow parallel
```

```
submission
/// of file opening operations expected to take a long time.
/// TFile::Open(TFileOpenHandle *) may block if the file is not
yet ready.
/// The sequence
///     TFile::Open(TFile::AsyncOpen(const char *, ...))
/// is equivalent to
///     TFile::Open(const char *, ...)
/// To be effective, the underlying TFile implementation must be
 able to
/// support asynchronous open functionality. Currently, only TXN
etFile
/// supports it. If the functionality is not implemented, this c
all acts
/// transparently by returning an handle with the arguments for
the
/// standard synchronous open run by TFile::Open(TFileOpenHandle
 *).
/// The retuned handle will be adopted by TFile after opening co
mpletion
/// in TFile::Open(TFileOpenHandle *); if opening is not finaliz
ed the
/// handle must be deleted by the caller.

   static TFile       *Open(const char *name, Option_t *option =
"",
                            const char *ftitle = "", Int_t compr
ess = 1,
                            Int_t netopt = 0);
/// Create / open a file
/// The type of the file can be either a
/// TFile, TNetFile, TWebFile or any TFile derived class for whi
ch an
/// plugin library handler has been registered with the plugin m
anager
/// (for the plugin manager see the TPluginManager class). The r
eturned
/// type of TFile depends on the file name specified by 'url'.
/// If 'url' is a '|'-separated list of file URLs, the 'URLs' ar
e tried
```

```
/// sequentially in the specified order until a successful open.
/// If the file starts with "root:", "roots:" or "rootk:" a TNet
File object
/// will be returned, with "http:" a TWebFile, with "file:" a lo
cal TFile,
/// etc. (see the list of TFile plugin handlers in $ROOTSYS/etc/
system.rootrc
/// for regular expressions that will be checked) and as last a
local file will
/// be tried.
/// Before opening a file via TNetFile a check is made to see if
 the URL
/// specifies a local file. If that is the case the file will be
 opened
/// via a normal TFile. To force the opening of a local file via
 a
/// TNetFile use either TNetFile directly or specify as host "lo
calhost".
/// The netopt argument is only used by TNetFile. For the meanin
g of the
/// options and other arguments see the constructors of the indi
vidual
/// file classes. In case of error returns 0.
/// For TFile implementations supporting asynchronous file open,
 see
/// TFile::AsyncOpen(...), it is possible to request a timeout w
ith the
/// option <b>TIMEOUT=<secs></b>: the timeout must be specified
in seconds and
/// it will be internally checked with granularity of one millis
ec.
/// For remote files there is the option: <b>CACHEREAD</b> opens
 an existing
/// file for reading through the file cache. The file will be do
wnloaded to
/// the cache and opened from there. If the download fails, it w
ill be opened remotely.
/// The file will be downloaded to the directory specified by Se
tCacheFileDir().
```

```
   static TFile       *Open(TFileOpenHandle *handle);
/// Waits for the completion of an asynchronous open request.
/// Returns the pointer to the associated TFile, transferring ow
nership of the
/// handle to the TFile instance.

   static EFileType    GetType(const char *name, Option_t *optio
n = "", TString *prefix = 0);
/// Resolve the file type as a function of the protocol field in
 'name'
/// If defined, the string 'prefix' is added when testing the lo
cality of
/// a 'name' with network-like structure (i.e. root://host//path
); if the file
/// is local, on return 'prefix' will contain the actual local p
ath of the file.

   static EAsyncOpenStatus GetAsyncOpenStatus(const char *name);
/// Get status of the async open request related to 'name'.

   static EAsyncOpenStatus GetAsyncOpenStatus(TFileOpenHandle *h
andle);
/// Get status of the async open request related to 'handle'.

   static const TUrl  *GetEndpointUrl(const char *name);
/// Get final URL for file being opened asynchronously.
/// Returns 0 is the information is not yet available.

   static Long64_t     GetFileBytesRead();
   static Long64_t     GetFileBytesWritten();
/// Static function returning the total number of bytes written
to all files.
/// Does not take into account what might still be in the write
caches.

   static Int_t        GetFileReadCalls();
/// Static function returning the total number of read calls fro
m all files.

   static Int_t        GetReadaheadSize();
```

```
/// Static function returning the readahead buffer size.

   static void         SetFileBytesRead(Long64_t bytes = 0);
   static void         SetFileBytesWritten(Long64_t bytes = 0);
   static void         SetFileReadCalls(Int_t readcalls = 0);
   static void         SetReadaheadSize(Int_t bufsize = 256000);
   static void         SetReadStreamerInfo(Bool_t readinfo=kTRUE)
;
/// Specify if the streamerinfos must be read at file opening.
/// If fgReadInfo is true (default) TFile::ReadStreamerInfo is c
alled
///  when opening the file.
/// It may be interesting to set fgReadInfo to false to speedup
the file
/// opening time or in case libraries containing classes referen
ced
/// by the file have not yet been loaded.
/// if fgReadInfo is false, one can still read the StreamerInfo
with
///    myfile.ReadStreamerInfo();

   static Bool_t       GetReadStreamerInfo();
/// If the streamerinfos are to be read at file opening.
/// See TFile::SetReadStreamerInfo for more documentation.

   static Long64_t     GetFileCounter();
   static void         IncrementFileCounter();

   static Bool_t       SetCacheFileDir(const char *cacheDir, Boo
l_t operateDisconnected = kTRUE,
                                       Bool_t forceCacheread = k
FALSE);
/// Sets the directory where to locally stage/cache remote files.

/// If the directory is not writable by us return kFALSE.

   static const char  *GetCacheFileDir();
/// Get the directory where to locally stage/cache remote files.

   static Bool_t       ShrinkCacheFileDir(Long64_t shrinkSize, L
```

```
ong_t cleanupInteval = 0);
/// Try to shrink the cache to the desired size.
/// With the clenupinterval you can specify the minimum amount o
f time after
/// the previous cleanup before the cleanup operation is repeate
d in
/// the cache directory

   static Bool_t       Cp(const char *src, const char *dst, Bool
_t progressbar = kTRUE,
                           UInt_t buffersize = 1000000);
/// Allows to copy file from src to dst URL. Returns kTRUE in ca
se of success,
/// kFALSE otherwise.

   static UInt_t       SetOpenTimeout(UInt_t timeout);  // in ms
/// Sets open timeout time (in ms). Returns previous timeout val
ue.

   static UInt_t       GetOpenTimeout(); // in ms
/// Returns open timeout (in ms).

   static Bool_t       SetOnlyStaged(Bool_t onlystaged);
   static Bool_t       GetOnlyStaged();
```

code

```
//打开文件
TFile f("demo.root");
if (f.IsZombie()) {
cout << "Error opening file" << endl;
exit(-1);}
else {
 .....
 }
```

```
//多个cycle时候读取某一部分数据
TTree *tt;
gFile->GetObject("NameOfTree;1",tt)
```

```
//Saving/Reading Histograms to/from a File
//The following statements create a ROOT file and store a histog
ram on the file. Because TH1 derives from TNamed , the key ident
ifier on the file is the histogram name:
TFile f("histos.root","new");
TH1F h1("hgaus","histo from a gaussian",100,-3,3);
h1.FillRandom("gaus",10000);
h1->Write();
//To read this histogram in another ROOT session, do:
TFile f("histos.root");
TH1F *h = (TH1F*)f.Get("hgaus");
One can save all histograms in memory to the file by:
file->Write();
For a more detailed explanation, see “Input/Output”.


TTree *fChain = (TTree*)gFile->Get("t");
Long64_t nentries = fChain->GetEntriesFast();
```

```
/// If the file is not usable (a zombie), you can test for this
case
/// with code like:
TFile f("myfile.root");
if (f.IsZombie()) {/*<actions to take if file is unusable>*/}

/// If the file has been recovered, the bit kRecovered is set in
 the TFile object in memory.
/// You can test if the file has been recovered with
if (f.TestBit(TFile::kRecovered)) {/*... the file has been recov
ered*/}
```

```
// function  File::Map()
```

```
/// List the contents of a file sequentially.
/// For each logical record found, it prints:
///
///     Date/Time  Record_Adress Logical_Record_Length  ClassNam
e  CompressionFactor
///
/// Example of output
///
///     20010404/150437  At:64        N=150       TFile
///     20010404/150440  At:214       N=28326     TBasket
 CX =  1.13
///     20010404/150440  At:28540     N=29616     TBasket
 CX =  1.08
///     20010404/150440  At:58156     N=29640     TBasket
 CX =  1.08
///     20010404/150440  At:87796     N=29076     TBasket
 CX =  1.10
///     20010404/150440  At:116872    N=10151     TBasket
 CX =  3.15
///     20010404/150441  At:127023    N=28341     TBasket
 CX =  1.13
///     20010404/150441  At:155364    N=29594     TBasket
 CX =  1.08
///     20010404/150441  At:184958    N=29616     TBasket
 CX =  1.08
///     20010404/150441  At:214574    N=29075     TBasket
 CX =  1.10
///     20010404/150441  At:243649    N=9583      TBasket
 CX =  3.34
///     20010404/150442  At:253232    N=28324     TBasket
 CX =  1.13
///     20010404/150442  At:281556    N=29641     TBasket
 CX =  1.08
///     20010404/150442  At:311197    N=29633     TBasket
 CX =  1.08
///     20010404/150442  At:340830    N=29091     TBasket
 CX =  1.10
///     20010404/150442  At:369921    N=10341     TBasket
 CX =  3.09
```

```
///     20010404/150442  At:380262    N=509       TH1F
 CX =  1.93
///     20010404/150442  At:380771    N=1769      TH2F
 CX =  4.32
///     20010404/150442  At:382540    N=1849      TProfile
 CX =  1.65
///     20010404/150442  At:384389    N=18434     TNtuple
 CX =  4.51
///     20010404/150442  At:402823    N=307       KeysList
///     20010404/150443  At:403130    N=4548      StreamerInfo
 CX =  3.65
///     20010404/150443  At:407678    N=86        FreeSegments
///     20010404/150443  At:407764    N=1         END
```

```
// function TFile::GetStreamerInfoList()

/// Read the list of TStreamerInfo objects written to this file.
/// The function returns a TList. It is the user's responsibility

/// to delete the list created by this function.
/// Using the list, one can access additional information, e.g.:

TFile f("myfile.root");
auto list = f.GetStreamerInfoList();
auto info = (TStreamerInfo*)list->FindObject("MyClass");
auto classversionid = info->GetClassVersion();
delete list;
```

example

```
//文本读取数据存ROOT文件：
#include "Riostream.h"
void basic() {
//  Read data from an ascii file and create a root file with an
histogram and an ntuple.
// this file has 3 columns of float data
```

```
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentM
acroName());
   std::cout<<dir<<std::endl;
   dir.ReplaceAll("basic.C","");
   dir.ReplaceAll("/./","/");std::cout<<dir<<std::endl;//当前目录
路径

   ifstream in;
   in.open(Form("%sbasic.dat",dir.Data()));

   Float_t x,y,z;
   Int_t nlines = 0;
   TFile *f = new TFile("basic.root","RECREATE");
   TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
   TNtuple *ntuple = new TNtuple("ntuple","data from ascii file",
"x:y:z");

   while (1) {
      in >> x >> y >> z;
      if (!in.good()) break;
      if (nlines < 5) printf("x=%8f, y=%8f, z=%8f\n",x,y,z);
      h1->Fill(x);
      ntuple->Fill(x,y,z);
      nlines++;
   }
   printf(" found %d points\n",nlines);

   in.close();

   f->Write();
}

void basic2() {
//   example of macro to create can ntuple reading data from an
ascii file.
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentM
acroName());
   dir.ReplaceAll("basic2.C","");
   dir.ReplaceAll("/./","/");
```

```
   TFile *f = new TFile("basic2.root","RECREATE");
   TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
   TTree *T = new TTree("ntuple","data from ascii file");
   Long64_t nlines = T->ReadFile(Form("%sbasic.dat",dir.Data()),
"x:y:z");
   printf(" found %lld points\n",nlines);
   T->Draw("x","z>2");
   T->Write();
}
```

TFileCacheWrite

TFileMerger

TFitter

TFitResult

TFitResultPtr

TFractionFitter

TGaxis

继承 TLine, TAttText

Service class for the graphical representation of axis.

Instances of this class are generated by the histograms and graphs painting classes when TAxis are drawn. TGaxis is the "painter class"of TAxis. Therefore it is mainly used via TAxis` even if is some occasion it can be used directly to draw an axis which is not part of a graph or an instance. For instance to draw an extra scale on a plot.

- Basic definition
- Definition with a function
- Logarithmic axis
- Blank axis
- Tick marks' orientation
- Tick marks' size
- Labels' positionning
- Labels' orientation
- Labels' position on tick marks
- Labels' format
- Alphanumeric labels
- Number of divisions optimisation
- Maximum Number of Digits for the axis labels
- Optional grid
- Time axis

Logarithmic axis

By default axis are linear. To define a TGaxis as logarithmic, it is enough to create it with the option "G".

When plotting an histogram or a graph the logarithmic scale can be set using:

- gPad->SetLogx(1); set the logarithmic scale on the X axis

- gPad->SetLogy(1); set the logarithmic scale on the Y axis

When the SetMoreLogLabels() method is called more labels are drawn when in logarithmic scale and there is a small number of decades (less than 3).

Tick marks' orientation

By default tick marks are drawn on the positive side of the axis, except for vertical axis for which the default is negative. The chop parameter allows to control the tick marks orientation:

- chopt = "+": tick marks are drawn on Positive side. (default)
- chopt ="-": tick mark are drawn on the negative side.
- chopt = "+-": tick marks are drawn on both sides of the axis.
- chopt = "U": Unlabelled axis, default is labeled.

Tick marks' size

By default, tick marks have a length equal to 3 per cent of the axis length. When the option "S" is specified, the length of the tick marks is equal to fTickSize*axis_length, where fTickSize may be set via TGaxis::SetTickSize.

When plotting an histogram **h** the tick marks size can be changed using:

- h->GetXaxis()->SetTickLength(0.02); set the tick length for the X axis
- gStyle->SetTickLength(0.02,"x"); set the tick length for the X axis of all histograms drawn after this instruction.

A good way to remove tick marks on an axis is to set the tick length to 0: h->GetXaxis()->SetTickLength(0.);

Labels' positionning

Labels are normally drawn on side opposite to tick marks. However the option "=" allows to draw them on the same side.

Labels' orientation

By default axis labels are drawn parallel to the axis. However if the axis is vertical then are drawn perpendicular to the axis.

Labels' position on tick marks

By default axis labels are centered on tick marks. However, for vertical axis, they are right adjusted. The chop parameter allows to control the labels' position on tick marks:

- chopt = "R": labels are Right adjusted on tick mark.(default is centered)
- chopt = "L": labels are Left adjusted on tick mark.
- chopt = "C": labels are Centered on tick mark.
- chopt = "M": In the Middle of the divisions.

Labels' format

Blank characters are stripped, and then the label is correctly aligned. the dot, if last character of the string, is also stripped, unless the option "." (a dot, or period) is specified. if SetDecimals(kTRUE) has been called all labels have the same number of decimals after the "." The same is true if gStyle->SetStripDecimals(kFALSE) has been called.

In the following, we have some parameters, like tick marks length and characters height (in percentage of the length of the axis (user's coordinates)) The default values are as follows:

- Primary tick marks: 3.0 %
- Secondary tick marks: 1.5 %
- Third order tick marks: .75 %
- Characters height for labels: 4%
- Labels offset: 1.0 %

By default, an exponent of the form 10^N is used when the label values are either all very small or very large. One can disable the exponent by calling axis.SetNoExponent(kTRUE).

TGaxis::SetExponentOffset(Float_t xoff, Float_t yoff, Option_t *axis) is static function to set X and Y offset of the axis 10^n notation. It is in % of the pad size. It can be negative. *axis* specifies which axis ("x" or/and "y"), default is "x" if axis = "xz" set the two axes

Alphanumeric labels

Axis labels can be any alphanumeric character strings. Such axis can be produced only with histograms because the labels'definition is stored in TAxis.

Because the alphanumeric labels are usually longer that the numeric labels, their size is by default equal to "0.66666 * the_numeric_labels_size".

Number of divisions optimisation

By default the number of divisions on axis is optimised to show a coherent labelling of the main tick marks. The number of division (ndiv) is a composite integer given by:

```
ndiv = N1 + 100*N2 + 10000*N3
```

- N1 = number of 1st divisions.
- N2 = number of 2nd divisions.
- N3 = number of 3rd divisions.

by default the value of N1, N2 and N3 are maximum values. After optimisation the real number of divisions will be smaller or equal to these value. If one wants to bypass the optimisation, the option "N" should be given when the TGaxis is created. The option "I" also act on the number of division as it will force an integer labelling of the axis.

On an histogram pointer **h** the number of divisions can be set in different ways:.

- Directly on the histogram. The following will set the number of division to 510 on the X axis of **h**. To avoid optimization the number of divisions should be negative (ie: -510);

```
h->SetNdivisions(510, "X");
```

- On the axis itself:

```
h->GetXaxis()->SetNdivisions(510, kTRUE);
```

The first parameter is the number of division. If it is negative of if the second parameter is kFALSE then the number of divisions is not optimised. And other signature is also allowed:

```
h->GetXaxis()->SetNdivisions(10, 5, 0, kTRUE);
```

Maximum Number of Digits for the axis labels

The static function TGaxis::SetMaxDigits sets the maximum number of digits permitted for the axis labels above which the notation with 10^N is used. For example, to accept 6 digits number like 900000 on an axis call TGaxis::SetMaxDigits(6). The default value is 5. fgMaxDigits must be greater than 0.

Optional grid

The option "W" allows to draw a grid on the primary tick marks. In case of a log axis, the grid is only drawn for the primary tick marks if the number of secondary and tertiary divisions is 0. SetGridLength() allows to define the length of the grid.

When plotting an histogram or a graph the grid can be set ON or OFF using:

- gPad->SetGridy(1); set the grid on the X axis
- gPad->SetGridx(1); set the grid on the Y axis
- gPad->SetGrid(1,1); set the grid on both axis.

Time axis

Axis labels may be considered as times, plotted in a defined time format. The format is set with SetTimeFormat(). The TGaxis minimum (wmin) and maximum (wmax) values are considered as two time values in seconds. The time axis will be spread around the time offset value (set with SetTimeOffset()). Actually it will go from "TimeOffset+wmin" to "TimeOffset+wmax"

Usually time axis are created automatically via histograms, but one may also want to draw a time axis outside an "histogram context". This can be done thanks to the option "T" of TGaxis.

class

```
   TGaxis();
   TGaxis(Double_t xmin,Double_t ymin,Double_t xmax,Double_t yma
x,
          Double_t wmin,Double_t wmax,Int_t ndiv=510, Option_t *
chopt="",
          Double_t gridlength = 0);
// Where:
// - xmin : X origin coordinate in user's coordinates space.
// - xmax : X end axis coordinate in user's coordinates space.
// - ymin : Y origin coordinate in user's coordinates space.
// - ymax : Y end axis coordinate in user's coordinates space.
// - wmin : Lowest value for the tick mark labels written on the
 axis.
// - wmax : Highest value for the tick mark labels written on th
e axis.
// - ndiv : Number of divisions.
//    - ndiv=N1 + 100*N2 + 10000*N3
//    - N1=number of 1st divisions.
//    - N2=number of 2nd divisions.
//    - N3=number of 3rd divisions. e.g.:
//       - ndiv=0 --> no tick marks.
//       - ndiv=2 --> 2 divisions, one tick mark in the middle o
f the axis.
// - chopt : Drawing options (see below).
// - gridlength: grid length on main tick marks.
```

```
   TGaxis(Double_t xmin,Double_t ymin,Double_t xmax,Double_t yma
x,
          const char *funcname, Int_t ndiv=510, Option_t *chopt=
"",
          Double_t gridlength = 0);
   virtual ~TGaxis();

   virtual void        AdjustBinSize(Double_t A1,  Double_t A2,
 Int_t nold
                                    ,Double_t &BinLow, Double_t
&BinHigh, Int_t &nbins, Double_t &BinWidth);
/// Internal method for axis labels optimisation. This method ad
justs the bining
/// of the axis in order to have integer values for the labels.
/// \param[in]  A1,A2            Old WMIN,WMAX
/// \param[out] binLow,binHigh   New WMIN,WMAX
/// \param[in]  nold             Old NDIV (primary divisions)
/// \param[out] nbins            New NDIV
/// \param[out] binWidth         Bin width

   virtual void        CenterLabels(Bool_t center=kTRUE);
/// If center = kTRUE axis labels are centered in the center of
the bin.
/// The default is to center on the primary tick marks.
/// This option does not make sense if there are more bins than
tick marks.

   virtual void        CenterTitle(Bool_t center=kTRUE);/// If c
enter = kTRUE axis title will be centered. The default is right
adjusted.
   virtual void        DrawAxis(Double_t xmin,Double_t ymin,Doub
le_t xmax,Double_t ymax,
                                Double_t wmin,Double_t wmax,Int_
t ndiv=510, Option_t *chopt="",
                                Double_t gridlength = 0);/// Dra
w this axis with new attributes.
   Float_t             GetGridLength() const   {return fGridLeng
th;}
   TF1                *GetFunction() const     {return fFunction
;}
```

```
   Int_t                GetLabelColor() const   {return fLabelCol
or;}
   Int_t                GetLabelFont() const    {return fLabelFon
t;}
   Float_t              GetLabelOffset() const  {return fLabelOff
set;}
   Float_t              GetLabelSize() const    {return fLabelSiz
e;}
   Float_t              GetTitleOffset() const  {return fTitleOff
set;}
   Float_t              GetTitleSize() const    {return fTitleSiz
e;}
   virtual const char *GetName() const  {return fName.Data();}
   virtual const char *GetOption() const {return fChopt.Data();}
   virtual const char *GetTitle() const {return fTitle.Data();}
   static Int_t         GetMaxDigits();/// Static function return
ing fgMaxDigits (See SetMaxDigits).
   Int_t                GetNdiv() const         {return fNdiv;}
   Double_t             GetWmin() const         {return fWmin;}
   Double_t             GetWmax()  const        {return fWmax;}
   Float_t              GetTickSize() const     {return fTickSize
;}
   virtual void         ImportAxisAttributes(TAxis *axis);/// Int
ernal method to import TAxis attributes to this TGaxis.
   void                 LabelsLimits(const char *label, Int_t &fi
rst, Int_t &last);/// Internal method to find first and last cha
racter of a label.
   virtual void         Paint(Option_t *chopt="");/// Draw this a
xis with its current attributes.
   virtual void         PaintAxis(Double_t xmin,Double_t ymin,Dou
ble_t xmax,Double_t ymax,
                                  Double_t &wmin,Double_t &wmax,I
nt_t &ndiv, Option_t *chopt="",
                                  Double_t gridlength = 0, Bool_t
 drawGridOnly = kFALSE);/// Control function to draw an axis.
   virtual void         Rotate(Double_t X,  Double_t Y,  Double_t
 CFI, Double_t SFI
                              ,Double_t XT, Double_t YT, Double_t
 &U,   Double_t &V);///    Internal method to rotate axis coordi
nates.
```

```
   virtual void        SavePrimitive(std::ostream &out, Option_t
 *option = "");/// Save primitive as a C++ statement(s) on outpu
t stream out
   void                SetFunction(const char *funcname="");///
Specify a function to map the axis values.
   void                SetOption(Option_t *option="");/// To set
 axis options.
   void                SetLabelColor(Int_t labelcolor) {fLabelCo
lor = labelcolor;} // *MENU*
   void                SetLabelFont(Int_t labelfont) {fLabelFont
 = labelfont;} // *MENU*
   void                SetLabelOffset(Float_t labeloffset) {fLab
elOffset = labeloffset;} // *MENU*
   void                SetLabelSize(Float_t labelsize) {fLabelSi
ze = labelsize;} // *MENU*
   static void         SetMaxDigits(Int_t maxd=5);
/// Static function to set `fgMaxDigits` for axis.`fgMaxDigits`
is
/// the maximum number of digits permitted for the axis labels a
bove which the
/// notation with 10^N is used.For example, to accept 6 digits n
umber like 900000
/// on an axis call `TGaxis::SetMaxDigits(6)`. The default value
 is 5.
/// `fgMaxDigits` must be greater than 0.

   virtual void        SetName(const char *name); // *MENU* ///
Change the name of the axis.
   virtual void        SetNdivisions(Int_t ndiv) {fNdiv = ndiv;}
// *MENU*
   virtual void        SetMoreLogLabels(Bool_t more=kTRUE);  //
*MENU*
/// Set the kMoreLogLabels bit flag. When this option is selecte
d more labels are
/// drawn when in logarithmic scale and there is a small number
of decades (less than 3).
/// Note that this option is automatically inherited from TAxis

   virtual void        SetNoExponent(Bool_t noExponent=kTRUE);
// *MENU*
```

```
/// Set the NoExponent flag. By default, an exponent of the form
 10^N is used
/// when the label values are either all very small or very larg
e. One can disable
/// the exponent by calling axis.SetNoExponent(kTRUE).

   virtual void        SetDecimals(Bool_t dot=kTRUE);  // *MENU*
/// Set the decimals flag. By default, blank characters are stri
pped, and then the
/// label is correctly aligned. The dot, if last character of th
e string, is also
/// stripped, unless this option is specified. One can disable t
he option by
/// calling `axis.SetDecimals(kTRUE)`.
/// Note the bit is set in fBits (as opposed to fBits2 in TAxis!)


   void                SetTickSize(Float_t ticksize) {fTickSize
= ticksize;} // *MENU*
   void                SetTickLength(Float_t ticklength) {SetTic
kSize(ticklength);}
   void                SetGridLength(Float_t gridlength) {fGridL
ength = gridlength;}
   void                SetTimeFormat(const char *tformat);
/// Change the format used for time plotting.
/// The format string for date and time use the same options as
the one used
/// in the standard strftime C function, i.e. :
/// for date :
///  - `%a` abbreviated weekday name
///  - `%b` abbreviated month name
///  - `%d` day of the month (01-31)
///  - `%m` month (01-12)
///  - `%y` year without century
/// for time :
///  - `%H` hour (24-hour clock)
///  - `%I` hour (12-hour clock)
///  - `%p` local equivalent of AM or PM
///  - `%M` minute (00-59)
///  - `%S` seconds (00-61)
```

```
///  - `%%` %

   void                SetTimeOffset(Double_t toffset, Option_t
*option="local");
   virtual void        SetTitle(const char *title=""); // *MENU*
 /// Change the title of the axis.
   void                SetTitleOffset(Float_t titleoffset=1) {fT
itleOffset = titleoffset;} // *MENU*
/// Change the time offset. If option = "gmt", set display mode
to GMT.

   void                SetTitleSize(Float_t titlesize) {fTitleSi
ze = titlesize;} // *MENU*
   void                SetTitleFont(Int_t titlefont) {SetTextFon
t(titlefont);} // *MENU*
   void                SetTitleColor(Int_t titlecolor) {SetTextC
olor(titlecolor);} // *MENU*
   void                SetWmin(Double_t wmin) {fWmin = wmin;}
   void                SetWmax(Double_t wmax) {fWmax = wmax;}
   static void         SetExponentOffset(Float_t xoff=0., Float_
t yoff=0., Option_t *axis="xy");
/// Static function to set X and Y offset of the axis 10^n notat
ion.
/// It is in % of the pad size. It can be negative.
/// axis specifies which axis ("x","y"), default = "x"
/// if axis="xz" set the two axes
```

code

```
// Instead of the wmin,wmax arguments of the normal definition,
the
// name of a TF1 function can be specified. This function will b
e used to
// map the user coordinates to the axis values and ticks.

   TCanvas *c2 = new TCanvas("c2","c2",10,10,700,500);

   gStyle->SetOptStat(0);

   TH2F *h2 = new TH2F("h","Axes",100,0,10,100,-2,2);
   h2->Draw();

   TF1 *f1=new TF1("f1","-x",-10,10);
   TGaxis *A1 = new TGaxis(0,2,10,2,"f1",510,"-");
   A1->SetTitle("axis with decreasing values");
   A1->Draw();

   TF1 *f2=new TF1("f2","exp(x)",0,2);
   TGaxis *A2 = new TGaxis(1,1,9,1,"f2");
   A2->SetTitle("exponential axis");
   A2->SetLabelSize(0.03);
   A2->SetTitleSize(0.03);
   A2->SetTitleOffset(1.2);
   A2->Draw();

   TF1 *f3=new TF1("f3","log10(x)",1,1000);
   TGaxis *A3 = new TGaxis(2,-2,2,0,"f3",505,"G");
   A3->SetTitle("logarithmic axis");
   A3->SetLabelSize(0.03);
   A3->SetTitleSize(0.03);
   A3->SetTitleOffset(1.2);
   A3->Draw();
   return c2;
```

```
   c1 = new TCanvas("c1","Examples of TGaxis",10,10,700,100);
   c1->Range(-10,-1,10,1);

   TGaxis *axis = new TGaxis(-8,0.,8,0.,-100000,150000,2405,"tS"
);
   axis->SetLabelSize(0.3);
   axis->SetTickSize(0.2);

   TDatime da(2003,02,28,12,00,00);
   axis->SetTimeOffset(da.Convert());
   axis->SetTimeFormat("%d-%m-%Y");
   axis->Draw();
   return c1;
```

example

```
// The example below generates various kind of axis.

   TCanvas *c1 = new TCanvas("c1","Examples of TGaxis",10,10,700,
500);

   c1->Range(-10,-1,10,1);

   TGaxis *axis1 = new TGaxis(-4.5,-0.2,5.5,-0.2,-6,8,510,"");
   axis1->SetName("axis1");
   axis1->Draw();

   TGaxis *axis2 = new TGaxis(-4.5,0.2,5.5,0.2,0.001,10000,510,"
G");
   axis2->SetName("axis2");
   axis2->Draw();

   TGaxis *axis3 = new TGaxis(-9,-0.8,-9,0.8,-8,8,50510,"");
   axis3->SetName("axis3");
   axis3->Draw();

   TGaxis *axis4 = new TGaxis(-7,-0.8,-7,0.8,1,10000,50510,"G");
```

```
   axis4->SetName("axis4");
   axis4->Draw();

   TGaxis *axis5 = new TGaxis(-4.5,-0.6,5.5,-0.6,1.2,1.32,80506,
"-+");
   axis5->SetName("axis5");
   axis5->SetLabelSize(0.03);
   axis5->SetTextFont(72);
   axis5->SetLabelOffset(0.025);

   axis5->Draw();

   TGaxis *axis6 = new TGaxis(-4.5,0.6,5.5,0.6,100,900,50510,"-"
);
   axis6->SetName("axis6");
   axis6->Draw();

   TGaxis *axis7 = new TGaxis(8,-0.8,8,0.8,0,9000,50510,"+L");
   axis7->SetName("axis7");
   axis7->SetLabelOffset(0.01);
   axis7->Draw();

   //one can make axis going top->bottom. However because of a l
ong standing
   //problem, the two x values should not be equal
   TGaxis *axis8 = new TGaxis(6.5,0.8,6.499,-0.8,0,90,50510,"-")
;
   axis8->SetName("axis8");
   axis8->Draw();
   return c1;
```

TGraph

继承 TNamed, TAttLine, TAttFill, TAttMarker

class

```
   // TGraph status bits
   enum {
      kClipFrame     = BIT(10),  // clip to the frame boundary
      kNotEditable   = BIT(18)   // bit set if graph is non edit
able
   };

   TGraph();/// Graph default constructor.
   TGraph(Int_t n);
/// Constructor with only the number of points set
/// the arrays x and y will be set later

   TGraph(Int_t n, const Int_t *x, const Int_t *y);/// Graph nor
mal constructor with ints.
   TGraph(Int_t n, const Float_t *x, const Float_t *y);/// Graph
 normal constructor with floats.
   TGraph(Int_t n, const Double_t *x, const Double_t *y);/// Gra
ph normal constructor with doubles.
   TGraph(const TGraph &gr);/// Copy constructor for this graph
   TGraph& operator=(const TGraph&);/// Equal operator for this
graph
   TGraph(const TVectorF &vx, const TVectorF &vy);
/// Graph constructor with two vectors of floats in input
/// A graph is build with the X coordinates taken from vx and Y
coord from vy
/// The number of points in the graph is the minimum of number o
f points
/// in vx and vy.

   TGraph(const TVectorD &vx, const TVectorD &vy);
/// Graph constructor with two vectors of doubles in input
```

```
/// A graph is build with the X coordinates taken from vx and Y
coord from vy
/// The number of points in the graph is the minimum of number o
f points
/// in vx and vy.

   TGraph(const TH1 *h);/// Graph constructor importing its para
meters from the TH1 object passed as argument
   TGraph(const TF1 *f, Option_t *option="");
/// Graph constructor importing its parameters from the TF1 obje
ct passed as argument
/// - if option =="" (default), a TGraph is created with points
computed
///                at the fNpx points of f.
/// - if option =="d", a TGraph is created with points computed
with the derivatives
///                at the fNpx points of f.
/// - if option =="i", a TGraph is created with points computed
with the integral
///                at the fNpx points of f.
/// - if option =="I", a TGraph is created with points computed
with the integral
///                at the fNpx+1 points of f and the integral is
 normalized to 1.

   TGraph(const char *filename, const char *format="%lg %lg", Op
tion_t *option="");
/// Graph constructor reading input from filename.
/// filename is assumed to contain at least two columns of numbe
rs.
/// the string format is by default "%lg %lg".
/// this is a standard c formatting for scanf. If columns of num
bers should be
/// skipped, a "%*lg" or "%*s" for each column can be added,
/// e.g. "%lg %*lg %lg" would read x-values from the first and y
-values from
/// the third column.
/// For files separated by a specific delimiter different from '
 ' and '\t' (e.g. ';' in csv files)
/// you can avoid using %*s to bypass this delimiter by explicit
```

```
ly specify the "option" argument,
/// e.g. option=" \t,;" for columns of figures separated by any
of these characters (' ', '\t', ',', ';')
/// used once (e.g. "1;1") or in a combined way (" 1;,;;  1").
/// Note in that case, the instantiation is about 2 times slower.


   virtual ~TGraph();/// Graph default destructor.

   virtual void          Apply(TF1 *f);
/// Apply function f to all the data points
/// f may be a 1-D function TF1 or 2-d function TF2
/// The Y values of the graph are replaced by the new values com
puted
/// using the function

   virtual void          Browse(TBrowser *b);/// Browse
   virtual Double_t      Chisquare(TF1 *f1, Option_t *option="")
const;
/// Return the chisquare of this graph with respect to f1.
/// The chisquare is computed as the sum of the quantity below a
t each point:
/// \f[
///   \frac{(y-f1(x))^{2}}{ey^{2}+(\frac{1}{2}(exl+exh)f1'(x))^{
2}}
/// \f]
/// where x and y are the graph point coordinates and f1'(x) is
the derivative of function f1(x).
/// This method to approximate the uncertainty in y because of t
he errors in x, is called
/// "effective variance" method.
/// In case of a pure TGraph, the denominator is 1.
/// In case of a TGraphErrors or TGraphAsymmErrors the errors ar
e taken
/// into account.
/// By default the range of the graph is used whatever function
range.
/// Use option "R" to use the function range

   static Bool_t         CompareArg(const TGraph* gr, Int_t left
```

```
, Int_t right);
/// Return kTRUE if point number "left"'s argument (angle with r
espect to positive
/// x-axis) is bigger than that of point number "right". Can be
used by Sort.

   static Bool_t         CompareX(const TGraph* gr, Int_t left,
Int_t right);/// Return kTRUE if fX[left] > fX[right]. Can be us
ed by Sort.
   static Bool_t         CompareY(const TGraph* gr, Int_t left,
Int_t right);/// Return kTRUE if fY[left] > fY[right]. Can be us
ed by Sort.
   static Bool_t         CompareRadius(const TGraph* gr, Int_t l
eft, Int_t right);
/// Return kTRUE if point number "left"'s distance to origin is
bigger than
/// that of point number "right". Can be used by Sort.

   virtual void          ComputeRange(Double_t &xmin, Double_t &
ymin, Double_t &xmax, Double_t &ymax) const; /// Compute the x/y
 range of the points in this graph
   virtual Int_t         DistancetoPrimitive(Int_t px, Int_t py)
;
/// Compute distance from point px,py to a graph.
///  Compute the closest distance of approach from point px,py t
o this line.
///  The distance is computed in pixels units.

   virtual void          Draw(Option_t *chopt="");
/// Draw this graph with its current attributes.
/// The options to draw a graph are described in TGraphPainter c
lass.

   virtual void          DrawGraph(Int_t n, const Int_t *x, const
 Int_t *y, Option_t *option="");/// Draw this graph with new att
ributes.
   virtual void          DrawGraph(Int_t n, const Float_t *x, co
nst Float_t *y, Option_t *option="");/// Draw this graph with ne
w attributes.
   virtual void          DrawGraph(Int_t n, const Double_t *x=0,
```

```
const Double_t *y=0, Option_t *option="");/// Draw this graph wi
th new attributes.
   virtual void          DrawPanel(); // *MENU* /// Display a pa
nel with all graph drawing options.
   virtual Double_t      Eval(Double_t x, TSpline *spline=0, Opt
ion_t *option="") const; //通过 x 插值找 y 值，提供多种插值算法：线性
插值、光滑插值等
/// Interpolate points in this graph at x using a TSpline
///  -if spline==0 and option="" a linear interpolation between
the two points
///   close to x is computed. If x is outside the graph range, a
 linear
///   extrapolation is computed.
///  -if spline==0 and option="S" a TSpline3 object is created u
sing this graph
///   and the interpolated value from the spline is returned.
///   the internally created spline is deleted on return.
///  -if spline is specified, it is used to return the interpola
ted value.

   virtual void          ExecuteEvent(Int_t event, Int_t px, Int
_t py);
/// Execute action corresponding to one event.
///  This member function is called when a graph is clicked with
 the locator
///  If Left button clicked on one of the line end points, this
point
///     follows the cursor until button is released.
///  if Middle button clicked, the line is moved parallel to its
elf
///     until the button is released.

   virtual void          Expand(Int_t newsize);/// If array size
s <= newsize, expand storage to 2*newsize.
   virtual void          Expand(Int_t newsize, Int_t step);
/// If graph capacity is less than newsize points then make arra
y sizes
/// equal to least multiple of step to contain newsize points.
/// Returns kTRUE if size was altered
```

```
   virtual TObject      *FindObject(const char *name) const;///
Search object named name in the list of functions
   virtual TObject      *FindObject(const TObject *obj) const;//
/ Search object obj in the list of functions

   virtual TFitResultPtr Fit(const char *formula ,Option_t *opti
on="" ,Option_t *goption="", Axis_t xmin=0, Axis_t xmax=0); // *
MENU*   返回0表示拟合正常
/// Fit this graph with function with name fname.
/// interface to TGraph::Fit(TF1 *f1...
/// fname is the name of an already predefined function created
by TF1 or TF2
/// Predefined functions such as gaus, expo and poln are automat
ically
/// created by ROOT.
/// fname can also be a formula, accepted by the linear fitter (
linear parts divided
/// by "++" sign), for example "x++sin(x)" for fitting "[0]*x+[1
]*sin(x)"

   virtual TFitResultPtr Fit(TF1 *f1 ,Option_t *option="" ,Optio
n_t *goption="", Axis_t xmin=0, Axis_t xmax=0);
////////////////////////////////////////////////////////////////
///////////////
/// Fit this graph with function f1.
///
/// f1 is an already predefined function created by TF1.
/// Predefined functions such as gaus, expo and poln are automat
ically
/// created by ROOT.
///
/// The list of fit options is given in parameter option.
///
/// option | description
/// -------|------------
/// "W" | Set all weights to 1; ignore error bars
/// "U" | Use a User specified fitting algorithm (via SetFCN)
/// "Q" | Quiet mode (minimum printing)
/// "V" | Verbose mode (default is between Q and V)
/// "E" | Perform better Errors estimation using Minos technique
```

```
/// "B" | User defined parameter settings are used for predefine
d functions like "gaus", "expo", "poln", "landau". Use this opti
on when you want to fix one or more parameters for these functio
ns.
/// "M" | More. Improve fit results. It uses the IMPROVE command
 of TMinuit (see TMinuit::mnimpr). This algorithm attempts to im
prove the found local minimum by searching for a better one.
/// "R" | Use the Range specified in the function range
/// "N" | Do not store the graphics function, do not draw
/// "0" | Do not plot the result of the fit. By default the fitt
ed function is drawn unless the option "N" above is specified.
/// "+" | Add this new fitted function to the list of fitted fun
ctions (by default, any previous function is deleted)
/// "C" | In case of linear fitting, do not calculate the chisqu
are (saves time)
/// "F" | If fitting a polN, use the minuit fitter
/// "EX0" | When fitting a TGraphErrors or TGraphAsymErrors do n
ot consider errors in the coordinate
/// "ROB" | In case of linear fitting, compute the LTS regressio
n coefficients (robust (resistant) regression), using the defaul
t fraction of good points "ROB=0.x" - compute the LTS regression
 coefficients, using 0.x as a fraction of good points
/// "S" |  The result of the fit is returned in the TFitResultPt
r (see below Access to the Fit Result)
///
/// When the fit is drawn (by default), the parameter goption ma
y be used
/// to specify a list of graphics options. See TGraphPainter for
 a complete
/// list of these options.
///
/// In order to use the Range option, one must first create a fu
nction
/// with the expression to be fitted. For example, if your graph
/// has a defined range between -4 and 4 and you want to fit a g
aussian
/// only in the interval 1 to 3, you can do:
///
///        TF1 *f1 = new TF1("f1","gaus",1,3);
///        graph->Fit("f1","R");
```

```
///
/// Who is calling this function:
///
/// Note that this function is called when calling TGraphErrors:
:Fit
/// or TGraphAsymmErrors::Fit ot TGraphBentErrors::Fit
/// See the discussion below on error calculation.
///
/// ## Linear fitting:
///   When the fitting function is linear (contains the "++" sig
n) or the fitting
///   function is a polynomial, a linear fitter is initialised.
///   To create a linear function, use the following syntax: lin
ear parts
///   separated by "++" sign.
///   Example: to fit the parameters of "[0]*x + [1]*sin(x)", cr
eate a
///    TF1 *f1=new TF1("f1", "x++sin(x)", xmin, xmax);
///   For such a TF1 you don't have to set the initial condition
s.
///   Going via the linear fitter for functions, linear in param
eters, gives a
///   considerable advantage in speed.
///
/// ## Setting initial conditions:
///
///   Parameters must be initialized before invoking the Fit fun
ction.
///   The setting of the parameter initial values is automatic f
or the
///   predefined functions : poln, expo, gaus, landau. One can h
owever disable
///   this automatic computation by specifying the option "B".
///   You can specify boundary limits for some or all parameters
 via
///
///        f1->SetParLimits(p_number, parmin, parmax);
///   If parmin>=parmax, the parameter is fixed
///   Note that you are not forced to fix the limits for all par
ameters.
```

```
///   For example, if you fit a function with 6 parameters, you
can do:
///
///     func->SetParameters(0,3.1,1.e-6,0.1,-8,100);
///     func->SetParLimits(4,-10,-4);
///     func->SetParLimits(5, 1,1);
///   With this setup, parameters 0->3 can vary freely.
///   Parameter 4 has boundaries [-10,-4] with initial value -8.
///   Parameter 5 is fixed to 100.
///
/// ## Fit range:
///
///   The fit range can be specified in two ways:
///     - specify rxmax > rxmin (default is rxmin=rxmax=0)
///     - specify the option "R". In this case, the function wil
l be taken
///       instead of the full graph range.
/// ## Changing the fitting function:
///   By default a chi2 fitting function is used for fitting a T
Graph.
///   The function is implemented in FitUtil::EvaluateChi2.
///   In case of TGraphErrors an effective chi2 is used (see bel
ow TGraphErrors fit)
///   To specify a User defined fitting function, specify option
 "U" and
///   call the following functions:
///      TVirtualFitter::Fitter(mygraph)->SetFCN(MyFittingFunctio
on)
///   where MyFittingFunction is of type:
///   extern void MyFittingFunction(Int_t &npar, Double_t *gin,
Double_t &f,
///                                 Double_t *u, Int_t flag);
/// ## TGraphErrors fit:
///   In case of a TGraphErrors object, when x errors are presen
t, the error along x,
///   is projected along the y-direction by calculating the func
tion at the points x-exlow and
///   x+exhigh. The chisquare is then computed as the sum of the
 quantity below at each point:
///
```

```
/// \f[
///   \frac{(y-f(x))^{2}}{ey^{2}+(\frac{1}{2}(exl+exh)f'(x))^{2}}

/// \f]
///
///   where x and y are the point coordinates, and f'(x) is the
derivative of the
///   function f(x).
///
///   In case the function lies below (above) the data point, ey
 is ey_low (ey_high).
///
///   thanks to Andy Haas (haas@yahoo.com) for adding the case w
ith TGraphAsymmErrors
///             University of Washington
///
///   The approach used to approximate the uncertainty in y beca
use of the
///   errors in x is to make it equal the error in x times the s
lope of the line.
///   The improvement, compared to the first method (f(x+ exhigh
) - f(x-exlow))/2
///   is of (error of x)**2 order. This approach is called "effe
ctive variance method".
///   This improvement has been made in version 4.00/08 by Anna
Kreshuk.
///   The implementation is provided in the function FitUtil::Ev
aluateChi2Effective
///
/// NOTE:
/// 1. By using the "effective variance" method a simple linear
regression
///    becomes a non-linear case, which takes several iterations
///    instead of 0 as in the linear case.
/// 2. The effective variance technique assumes that there is no
 correlation
///    between the x and y coordinate.
/// 3. The standard chi2 (least square) method without error in
the coordinates (x) can
///    be forced by using option "EX0"
```

```
/// 4. The linear fitter doesn't take into account the errors in
 x. When fitting a
///    TGraphErrors with a linear functions the errors in x will
 not be considered.
///    If errors in x are important, go through minuit (use opti
on "F" for polynomial fitting).
/// 5. When fitting a TGraph (i.e. no errors associated with eac
h point),
///    a correction is applied to the errors on the parameters w
ith the following
///    formula: errorp *= sqrt(chisquare/(ndf-1))
///
///  ## Access to the fit result
///  The function returns a TFitResultPtr which can hold a  poin
ter to a TFitResult object.
///  By default the TFitResultPtr contains only the status of th
e fit which is return by an
///  automatic conversion of the TFitResultPtr to an integer. On
e can write in this case
///  directly:
///
///      Int_t fitStatus =  h->Fit(myFunc)
///
///  If the option "S" is instead used, TFitResultPtr contains t
he TFitResult and behaves
///  as a smart pointer to it. For example one can do:
///
///      TFitResultPtr r = h->Fit(myFunc,"S");
///      TMatrixDSym cov = r->GetCovarianceMatrix();  //  to acc
ess the covariance matrix
///      Double_t chi2   = r->Chi2(); // to retrieve the fit chi2

///      Double_t par0   = r->Value(0); // retrieve the value fo
r the parameter 0
///      Double_t err0   = r->ParError(0); // retrieve the error
 for the parameter 0
///      r->Print("V");     // print full information of fit inc
luding covariance matrix
///      r->Write();        // store the result in a file
///  The fit parameters, error and chi2 (but not covariance matr
```

```
ix) can be retrieved also
///  from the fitted function.
///  If the histogram is made persistent, the list of
///  associated functions is also persistent. Given a pointer (s
ee above)
///  to an associated function myfunc, one can retrieve the func
tion/fit
///  parameters with calls such as:
///      Double_t chi2 = myfunc->GetChisquare();
///      Double_t par0 = myfunc->GetParameter(0); //value of 1st
 parameter
///      Double_t err0 = myfunc->GetParError(0);  //error on fir
st parameter
/// ## Access to the fit status
///  The status of the fit can be obtained converting the TFitRe
sultPtr to an integer
///  independently if the fit option "S" is used or not:
///       TFitResultPtr r = h->Fit(myFunc,opt);
///       Int_t fitStatus = r;
///  The fitStatus is 0 if the fit is OK (i.e. no error occurred
).
///  The value of the fit status code is negative in case of an
error not connected with the
///  minimization procedure, for example when a wrong function i
s used.
///  Otherwise the return value is the one returned from the min
imization procedure.
///  When TMinuit (default case) or Minuit2 are used as minimize
r the status returned is :
///  fitStatus =  migradResult + 10*minosResult + 100*hesseResul
t + 1000*improveResult.
///  TMinuit will return 0 (for migrad, minos, hesse or improve)
 in case of success and 4 in
///  case of error (see the documentation of TMinuit::mnexcm). S
o for example, for an error
///  only in Minos but not in Migrad a fitStatus of 40 will be r
eturned.
///  Minuit2 will return also 0 in case of success and different
 values in migrad, minos or
///  hesse depending on the error.   See in this case the docume
```

```
ntation of
///  Minuit2Minimizer::Minimize for the migradResult, Minuit2Min
imizer::GetMinosError for the
///  minosResult and Minuit2Minimizer::Hesse for the hesseResult.

///  If other minimizers are used see their specific documentati
on for the status code
///  returned. For example in the case of Fumili, for the status
 returned see TFumili::Minimize.
///
/// ## Associated functions:
///   One or more object (typically a TF1*) can be added to the
list
///   of functions (fFunctions) associated with each graph.
///   When TGraph::Fit is invoked, the fitted function is added
to this list.
///   Given a graph gr, one can retrieve an associated function
///   with:  TF1 *myfunc = gr->GetFunction("myfunc");
///
///   If the graph is made persistent, the list of associated fu
nctions is also
///   persistent. Given a pointer (see above) to an associated f
unction myfunc,
///   one can retrieve the function/fit parameters with calls su
ch as:
///
///       Double_t chi2 = myfunc->GetChisquare();
///       Double_t par0 = myfunc->GetParameter(0); //value of 1s
t parameter
///       Double_t err0 = myfunc->GetParError(0);  //error on fi
rst parameter
///
/// ## Fit Statistics
///   You can change the statistics box to display the fit param
eters with
///   the TStyle::SetOptFit(mode) method. This mode has four dig
its.
///   mode = pcev  (default = 0111)
///
///       v = 1;  print name/values of parameters
```

```
///       e = 1;  print errors (if e=1, v must be 1)
///       c = 1;  print Chisquare/Number of degrees of freedom
///       p = 1;  print Probability
///
///   For example: gStyle->SetOptFit(1011);
///   prints the fit probability, parameter names/values, and er
rors.
///   You can change the position of the statistics box with the
se lines
///   (where g is a pointer to the TGraph):
///       Root > TPaveStats *st = (TPaveStats*)g->GetListOfFunct
ions()->FindObject("stats")
///       Root > st->SetX1NDC(newx1); //new x start position
///       Root > st->SetX2NDC(newx2); //new x end position

   virtual void          FitPanel(); // *MENU*
/// Display a GUI panel with all graph fit options.
/// See class TFitEditor for example

   Bool_t                GetEditable() const; /// Return kTRUE i
f kNotEditable bit is not set, kFALSE otherwise.
   TF1                  *GetFunction(const char *name) const;//
获得拟合函数，由拟合函数可得到拟合参数
/// Return pointer to function with name.
/// Functions such as TGraph::Fit store the fitted function in t
he list of
/// functions of this graph.

   TH1F                 *GetHistogram() const;
/// Returns a pointer to the histogram used to draw the axis
/// Takes into account the two following cases.
///  1. option 'A' was specified in TGraph::Draw. Return fHistog
ram
///  2. user had called TPad::DrawFrame. return pointer to hfram
e histogram

   TList                *GetListOfFunctions() const { return fFu
nctions; }
   virtual Double_t      GetCorrelationFactor() const;/// Return
 graph correlation factor
```

```
   virtual Double_t      GetCovariance() const;/// Return covari
ance of vectors x,y
   virtual Double_t      GetMean(Int_t axis=1) const;/// Return
mean value of X (axis=1)  or Y (axis=2)
   virtual Double_t      GetRMS(Int_t axis=1) const; /// Return
RMS of X (axis=1)  or Y (axis=2)
   Int_t                 GetMaxSize() const {return fMaxSize;}
   Int_t                 GetN() const {return fNpoints;}//获得填
充点个数
   virtual Double_t      GetErrorX(Int_t bin) const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
GraphErrors

   virtual Double_t      GetErrorY(Int_t bin) const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
GraphErrors

   virtual Double_t      GetErrorXhigh(Int_t bin) const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
GraphErrors
/// and TGraphAsymmErrors

   virtual Double_t      GetErrorXlow(Int_t bin)  const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
GraphErrors
/// and TGraphAsymmErrors

   virtual Double_t      GetErrorYhigh(Int_t bin) const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
GraphErrors
/// and TGraphAsymmErrors

   virtual Double_t      GetErrorYlow(Int_t bin)  const;
/// This function is called by GraphFitChisquare.
/// It always returns a negative value. Real implementation in T
```

```
GraphErrors
/// and TGraphAsymmErrors

   Double_t             *GetX()  const {return fX;}
   Double_t             *GetY()  const {return fY;}
   virtual Double_t     *GetEX() const {return 0;}
   virtual Double_t     *GetEY() const {return 0;}
   virtual Double_t     *GetEXhigh() const {return 0;}
   virtual Double_t     *GetEXlow()  const {return 0;}
   virtual Double_t     *GetEYhigh() const {return 0;}
   virtual Double_t     *GetEYlow()  const {return 0;}
   virtual Double_t     *GetEXlowd()  const {return 0;}
   virtual Double_t     *GetEXhighd() const {return 0;}
   virtual Double_t     *GetEYlowd()  const {return 0;}
   virtual Double_t     *GetEYhighd() const {return 0;}
   Double_t              GetMaximum()  const {return fMaximum;}
   Double_t              GetMinimum()  const {return fMinimum;}
   TAxis                *GetXaxis() const ; /// Get x axis of th
e graph.
   TAxis                *GetYaxis() const ; /// Get y axis of th
e graph.
   virtual Int_t         GetPoint(Int_t i, Double_t &x, Double_t
 &y) const;
/// Get x and y values for point number i.
/// The function returns -1 in case of an invalid request or the
 point number otherwise

   virtual void          InitExpo(Double_t xmin=0, Double_t xmax=
0); /// Compute Initial values of parameters for an exponential.
   virtual void          InitGaus(Double_t xmin=0, Double_t xmax=
0); /// Compute Initial values of parameters for a gaussian.
   virtual void          InitPolynom(Double_t xmin=0, Double_t x
max=0); /// Compute Initial values of parameters for a polynom.
   virtual Int_t         InsertPoint(); // *MENU* /// Insert a n
ew point at the mouse position
   virtual Double_t      Integral(Int_t first=0, Int_t last=-1)
const;
/// Integrate the TGraph data within a given (index) range.
/// Note that this function computes the area of the polygon enc
losed by the points of the TGraph.
```

```
/// The polygon segments, which are defined by the points of the
 TGraph, do not need to form a closed polygon,
/// since the last polygon segment, which closes the polygon, is
 taken as the line connecting the last TGraph point
/// with the first one. It is clear that the order of the point
is essential in defining the polygon.
/// Also note that the segments should not intersect.
/// NB:
///  - if last=-1 (default) last is set to the last point.
///  - if (first <0) the first point (0) is taken.
/// ### Method:
/// There are many ways to calculate the surface of a polygon. I
t all depends on what kind of data
/// you have to deal with. The most evident solution would be to
 divide the polygon in triangles and
/// calculate the surface of them. But this can quickly become c
omplicated as you will have to test
/// every segments of every triangles and check if they are inte
rsecting with a current polygon's
/// segment or if it goes outside the polygon. Many calculations
 that would lead to many problems...
/// ### The solution (implemented by R.Brun)
/// Fortunately for us, there is a simple way to solve this prob
lem, as long as the polygon's
/// segments don't intersect.
/// It takes the x coordinate of the current vertex and multiply
 it by the y coordinate of the next
/// vertex. Then it subtracts from it the result of the y coordi
nate of the current vertex multiplied
/// by the x coordinate of the next vertex. Then divide the resu
lt by 2 to get the surface/area.
/// ### Sources
///  - http://forums.wolfram.com/mathgroup/archive/1998/Mar/msg0
0462.html
///  - http://stackoverflow.com/questions/451426/how-do-i-calcul
ate-the-surface-area-of-a-2d-polygon


   virtual Bool_t        IsEditable() const {return !TestBit(kNo
tEditable);}
```

```
   virtual Int_t         IsInside(Double_t x, Double_t y) const;
//判断（x，y）是否在TCut选定的范围
/// Return 1 if the point (x,y) is inside the polygon defined by
/// the graph vertices 0 otherwise.
/// Algorithm:
/// The loop is executed with the end-point coordinates of a lin
e segment
/// (X1,Y1)-(X2,Y2) and the Y-coordinate of a horizontal line.
/// The counter inter is incremented if the line (X1,Y1)-(X2,Y2)
 intersects
/// the horizontal line. In this case XINT is set to the X-coord
inate of the
/// intersection point. If inter is an odd number, then the poin
t x,y is within
/// the polygon.

   virtual void          LeastSquareFit(Int_t m, Double_t *a, Do
uble_t xmin=0, Double_t xmax=0);
/// Least squares polynomial fitting without weights.
/// \param [in] m     number of parameters
/// \param [in] ma     array of parameters
/// \param [in] mfirst 1st point number to fit (default =0)
/// \param [in] mlast  last point number to fit (default=fNpoint
s-1)
/// based on CERNLIB routine LSQ: Translated to C++ by Rene Brun

   virtual void          LeastSquareLinearFit(Int_t n, Double_t
&a0, Double_t &a1, Int_t &ifail, Double_t xmin=0, Double_t xmax=0
);
/// Least square linear fit without weights.
///  Fit a straight line (a0 + a1*x) to the data in this graph.
/// \param [in] ndata        if ndata<0, fits the logarithm of t
he graph (used in InitExpo() to set
///                          the initial parameter values for a
fit with exponential function.
/// \param [in] a0           constant
/// \param [in] a1           slope
/// \param [in] ifail        return parameter indicating the sta
tus of the fit (ifail=0, fit is OK)
/// \param [in] xmin, xmax   fitting range
```

```
///  extracted from CERNLIB LLSQ: Translated to C++ by Rene Brun

   virtual Int_t         Merge(TCollection* list);
/// Adds all graphs from the collection to this graph.
/// Returns the total number of poins in the result or -1 in cas
e of an error.

   virtual void          Paint(Option_t *chopt="");/// Draw this
 graph with its current attributes.
   void                  PaintGraph(Int_t npoints, const Double_
t *x, const Double_t *y, Option_t *chopt);/// Draw the (x,y) as
a graph.
   void                  PaintGrapHist(Int_t npoints, const Doub
le_t *x, const Double_t *y, Option_t *chopt);/// Draw the (x,y)
as a histogram.
   virtual void          PaintStats(TF1 *fit);/// Draw the stats
   virtual void          Print(Option_t *chopt="") const;/// Pri
nt graph values.
   virtual void          RecursiveRemove(TObject *obj); /// Recu
rsively remove object from the list of functions
   virtual Int_t         RemovePoint(); // *MENU* /// Delete poi
nt close to the mouse position
   virtual Int_t         RemovePoint(Int_t ipoint); /// Delete p
oint number ipoint
   virtual void          SavePrimitive(std::ostream &out, Option
_t *option = ""); /// Save primitive as a C++ statement(s) on ou
tput stream out
   virtual void          SetEditable(Bool_t editable=kTRUE); //
*TOGGLE* *GETTER=GetEditable
/// if editable=kFALSE, the graph cannot be modified with the mo
use
///  by default a TGraph is editable

   virtual void          SetHistogram(TH1F *h) {fHistogram = h;}
   virtual void          SetMaximum(Double_t maximum=-1111); //
*MENU* /// Set the maximum of the graph.
   virtual void          SetMinimum(Double_t minimum=-1111); //
*MENU* /// Set the minimum of the graph.
   virtual void          Set(Int_t n);
/// Set number of points in the graph
```

```
/// Existing coordinates are preserved
/// New coordinates above fNpoints are preset to 0.

   virtual void          SetPoint(Int_t i, Double_t x, Double_t
y);/// Set x and y values for point number i. 逐个点填充，i从0开始
   virtual void          SetTitle(const char *title="");    // *
MENU* /// Set graph title.
   virtual void          Sort(Bool_t (*greater)(const TGraph*, I
nt_t, Int_t)=&TGraph::CompareX,
                                Bool_t ascending=kTRUE, Int_t low=0
, Int_t high=-1111);
/// Sorts the points of this TGraph using in-place quicksort (se
e e.g. older glibc).
/// To compare two points the function parameter greaterfunc is
used (see TGraph::CompareX for an
/// example of such a method, which is also the default comparis
on function for Sort). After
/// the sort, greaterfunc(this, i, j) will return kTRUE for all
i>j if ascending == kTRUE, and
/// kFALSE otherwise.
/// The last two parameters are used for the recursive quick sor
t, stating the range to be sorted

   virtual void          UseCurrentStyle();
/// Set current style settings in this graph
/// This function is called when either TCanvas::UseCurrentStyle
/// or TROOT::ForceStyle have been invoked.

   void                  Zero(Int_t &k,Double_t AZ,Double_t BZ,D
ouble_t E2,Double_t &X,Double_t &Y,Int_t maxiterations);
/// Find zero of a continuous function.
/// This function finds a real zero of the continuous real
/// function Y(X) in a given interval (A,B). See accompanying
/// notes for details of the argument list and calling sequence
```

code

```
// sort points along x axis
graph->Sort();
// sort points along their distance to origin
graph->Sort(&TGraph::CompareRadius);

Bool_t CompareErrors(const TGraph* gr, Int_t i, Int_t j) {
  const TGraphErrors* ge=(const TGraphErrors*)gr;
  return (ge->GetEY()[i]>ge->GetEY()[j]); }
// sort using the above comparison function, largest errors first

graph->Sort(&CompareErrors, kFALSE);
```

example

```
const Int_t n = 20;
Double_t x[n], y[n];
for (Int_t i=0;i<n;i++) {
x[i] = i*0.1;
y[i] = 10*sin(x[i]+0.2);}
// create graph
TGraph *gr = new TGraph(n,x,y);
TCanvas *c1 = new TCanvas("c1","Graph Draw Options",200,10,600,4
00);
// draw the graph with axis, continuous line, and put a * at eac
h point
gr->Draw("AC*");
```

```
//Draw a graph with text attached to each point.

const Int_t n = 10;
TGraph *gr = new TGraph(n);
gr->SetTitle("A Simple Graph Example with Text");
gr->SetMarkerStyle(20);
TExec *ex = new TExec("ex","drawtext();");
gr->GetListOfFunctions()->Add(ex);
……

//然后加上以下代码：
void drawtext()
{
   Int_t i,n;
   Double_t x,y;
   TLatex *l;

   TGraph *g = (TGraph*)gPad->GetListOfPrimitives()->FindObject(
"Graph");
   n = g->GetN();
   for (i=1; i<n; i++) {
      g->GetPoint(i,x,y);
      l = new TLatex(x,y+0.2,Form("%4.2f",y));
      l->SetTextSize(0.025);
      l->SetTextFont(42);
      l->SetTextAlign(21);
      l->Paint();
   }
}
```

```
//Draw a simple graph。

TCanvas *c1 = new TCanvas("c1","A Simple Graph Example",200,10,7
00,500);
c1->SetFillColor(42);
c1->SetGrid();

const Int_t n = 20;
Double_t x[n], y[n];
for (Int_t i=0;i<n;i++) {
  x[i] = i*0.1;
  y[i] = 10*sin(x[i]+0.2);
  printf(" i %i %f %f \n",i,x[i],y[i]);
 }
gr = new TGraph(n,x,y);
gr->SetLineColor(2);
gr->SetLineWidth(4);
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->SetTitle("a simple graph");//设置标题
gr->GetXaxis()->SetTitle("X title");//设置横坐标
gr->GetYaxis()->SetTitle("Y title");//设置纵坐标
gr->Draw("ACP");
```

```
//Draw several graphs with an exclusion zones.一个画板上画出多个Gra
ph，图片上加注释。

TCanvas *c = new TCanvas("c","Charged Higgs L300 Contour",0,0,700
,700);
c->SetTickx();
c->SetTicky();
c->SetGridx();
c->SetGridy();

TH1 *frame = new TH1F("frame","",1000,50,500);
frame->SetMinimum(1);
frame->SetMaximum(50);
frame->SetDirectory(0);
```

```
frame->SetStats(0);
frame->GetXaxis()->SetTitle("m_{A} (GeV)");
frame->GetXaxis()->SetTickLength(0.02);
frame->GetXaxis()->SetLabelSize(0.03);
frame->GetYaxis()->SetTitle("tan#beta");
frame->GetYaxis()->SetMoreLogLabels();
frame->GetYaxis()->SetLabelSize(0.03);
frame->Draw(" ");
c->SetLogy();

TGraph *gr1 = new TGraph(10);
gr1->SetFillColor(6);
gr1->SetFillStyle(3005);
gr1->SetLineColor(6);
gr1->SetLineWidth(603);
gr1->SetPoint(0,140,0.5);
gr1->SetPoint(1,130,2.9);
gr1->SetPoint(2,124.677,3.83726);
gr1->SetPoint(3,113.362,6.06903);
gr1->SetPoint(4,108.513,8.00221);
gr1->SetPoint(5,111.746,10.0272);
gr1->SetPoint(6,119.828,12.8419);
gr1->SetPoint(7,135.991,30.0872);
gr1->SetPoint(8,140,40);
gr1->SetPoint(9,135,60);
gr1->Draw("C");
TLatex *tex = new TLatex(140.841,37.9762,
             "#leftarrow t #rightarrow bH^{+}, H^{+} #rightarrow
 #tau#nu");
tex->SetTextColor(6);
tex->Draw();

TGraph *gr2 = new TGraph(15);
gr2->SetName("Graph");
gr2->SetTitle("Graph");
gr2->SetFillColor(1);
gr2->SetFillStyle(3005);
gr2->SetLineWidth(3);
gr2->SetPoint(0,499.192,3.02622);
gr2->SetPoint(1,427.748,3.06233);
```

```
gr2->SetPoint(2,358.244,3.10722);
gr2->SetPoint(3,305.711,3.24589);
gr2->SetPoint(4,244.289,3.36617);
gr2->SetPoint(5,206.304,3.7544);
gr2->SetPoint(6,178.017,4.50347);
gr2->SetPoint(7,148.114,6.20297);
gr2->SetPoint(8,131.142,8.00221);
gr2->SetPoint(9,111.746,8.48188);
gr2->SetPoint(10,102.047,9.52921);
gr2->SetPoint(11,96.3901,13.2212);
gr2->SetPoint(12,92.3491,19.0232);
gr2->SetPoint(13,90.7328,26.3935);
gr2->SetPoint(14,93.1573,50.4385);
gr2->Draw("L");
tex = new TLatex(346.929,6.62281,"ATLAS");
tex->SetLineWidth(2);
tex->Draw();
tex = new TLatex(328.341,5.24703,"#intLdt = 300 fb^{-1}");
tex->SetTextSize(0.0297619);
tex->SetLineWidth(2);
tex->Draw();
tex = new TLatex(340.463,4.1874,"Maximal mixing");
tex->SetTextSize(0.0297619);
tex->SetLineWidth(2);
tex->Draw();
tex = new TLatex(413.2,2.51608,"LEP 2000");
tex->SetTextSize(0.0297619);
tex->SetLineWidth(2);
tex->Draw();

TGraph *gr3 = new TGraph(10);
gr3->SetName("Graph");
gr3->SetTitle("Graph");
gr3->SetFillColor(2);
gr3->SetFillStyle(3004);
gr3->SetLineColor(2);
gr3->SetLineWidth(603);
gr3->SetPoint(0,176.84,10.7499);
gr3->SetPoint(1,190.575,11.9912);
gr3->SetPoint(2,211.58,12.7108);
```

```
gr3->SetPoint(3,243.088,12.3457);
gr3->SetPoint(4,279.443,12.6185);
gr3->SetPoint(5,302.065,12.9916);
gr3->SetPoint(6,331.957,13.7713);
gr3->SetPoint(7,369.928,14.2821);
gr3->SetPoint(8,425.673,16.1651);
gr3->SetPoint(9,499.192,18.1635);
gr3->Draw("C");
tex = new TLatex(188.151,9.36035,
         "gb #rightarrow tH^{+}, H^{+} #rightarrow #tau#nu");
tex->SetTextColor(2);
tex->Draw();

TGraph *gr4 = new TGraph(10);
gr4->SetName("Graph");
gr4->SetTitle("Graph");
gr4->SetFillColor(4);
gr4->SetFillStyle(3004);
gr4->SetLineColor(4);
gr4->SetLineWidth(-603);
gr4->SetPoint(0,178.456,2.91797);
gr4->SetPoint(1,200.269,3.40033);
gr4->SetPoint(2,229.354,3.96243);
gr4->SetPoint(3,249.551,4.07959);
gr4->SetPoint(4,269.749,3.71097);
gr4->SetPoint(5,298.025,3.09308);
gr4->SetPoint(6,341.652,2.89679);
gr4->SetPoint(7,378.007,2.57808);
gr4->SetPoint(8,441.023,2.16454);
gr4->SetPoint(9,499.677,1.76145);
gr4->Draw("C");
tex = new TLatex(165.,1.15498,
         "gb #rightarrow tH^{+}, H^{+} #rightarrow tb");
tex->SetTextColor(4);
tex->Draw();

TGraph *gr5 = new TGraph(10);
gr5->SetName("Graph");
gr5->SetTitle("Graph");
gr5->SetFillColor(4);
```

```
gr5->SetFillStyle(3004);
gr5->SetLineColor(4);
gr5->SetLineWidth(603);
gr5->SetPoint(0,152.603,23.0996);
gr5->SetPoint(1,188.151,18.8373);
gr5->SetPoint(2,239.048,15.2499);
gr5->SetPoint(3,264.901,15.8156);
gr5->SetPoint(4,299.641,18.8373);
gr5->SetPoint(5,334.381,20.7085);
gr5->SetPoint(6,360.233,22.4362);
gr5->SetPoint(7,396.589,24.4859);
gr5->SetPoint(8,433.752,25.7669);
gr5->SetPoint(9,499.192,27.3132);
gr5->Draw("C");
```

TGraph2D

TGraph2DErrors

TGraphErrors

class

code

example

```
//Draw a graph with error bars 带误差棒的图

TCanvas *c1 = new TCanvas("c1","A Simple Graph with error bars",
200,10,700,500);

c1->SetFillColor(42);
c1->SetGrid();
c1->GetFrame()->SetFillColor(21);
c1->GetFrame()->SetBorderSize(12);

const Int_t n = 10;
Float_t x[n]  = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61,0.7,0.85,0.89
,0.95};
Float_t y[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
Float_t ex[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
Float_t ey[n] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
TGraphErrors *gr = new TGraphErrors(n,x,y,ex,ey);
gr->SetTitle("TGraphErrors Example");
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->Draw("ALP");
```

TGraphPolar

class

code

example

```
TCanvas *CPol = new TCanvas("CPol","TGraphPolar Examples",700,700
);
Double_t rmin=0;
Double_t rmax=TMath::Pi()*2;
Double_t r[1000];
Double_t theta[1000];
TF1 * fp1 = new TF1("fplot","cos(x)",rmin,rmax);
for (Int_t ipt = 0; ipt < 1000; ipt++) {
r[ipt] = ipt*(rmax-rmin)/1000+rmin;
theta[ipt] = fp1->Eval(r[ipt]);}
TGraphPolar * grP1 = new TGraphPolar(1000,r,theta);
grP1->SetLineColor(2);
grP1->Draw("AOL");
```

TH1

- TH1C : histograms with one byte per channel. Maximum bin content = 127
- TH1S : histograms with one short per channel. Maximum bin content = 32767
- TH1I : histograms with one int per channel. Maximum bin content = 2147483647
- TH1F : histograms with one float per channel. Maximum precision 7 digits
- TH1D : histograms with one double per channel. Maximum precision 14 digits

```
The TH*C classes also inherit from the array class TArrayC.
The TH*S classes also inherit from the array class TArrayS.
The TH*I classes also inherit from the array class TArrayI.
The TH*F classes also inherit from the array class TArrayF.
The TH*D classes also inherit from the array class TArrayD.
```

When an histogram is created, a reference to it is automatically added to the list of in-memory objects for the current file or directory. This default behaviour can be changed by:

```
h->SetDirectory(0);          // for the current histogram h
TH1::AddDirectory(kFALSE);   // sets a global switch disabling t
he reference
```

When the histogram is deleted, the reference to it is removed from the list of objects in memory. When a file is closed, all histograms in memory associated with this file are automatically deleted.

All histogram types support either fix or variable bin sizes. 2-D histograms may have fix size bins along X and variable size bins along Y or vice-versa. The functions to fill, manipulate, draw or access histograms are identical in both cases.

Convention for numbering bins

For all histogram types: nbins, xlow, xup

```
bin = 0;        underflow bin
bin = 1;        first bin with low-edge xlow INCLUDED
bin = nbins;    last bin with upper-edge xup EXCLUDED
bin = nbins+1; overflow bin
```

class

```
protected:
   TH1();
   TH1(const char *name,const char *title,Int_t nbinsx,Double_t
xlow,Double_t xup);
   TH1(const char *name,const char *title,Int_t nbinsx,const Flo
at_t *xbins);
   TH1(const char *name,const char *title,Int_t nbinsx,const Dou
ble_t *xbins);
   virtual Int_t    BufferFill(Double_t x, Double_t w);
   virtual Bool_t   FindNewAxisLimits(const TAxis* axis, const D
ouble_t point, Double_t& newMin, Double_t &newMax);
   virtual void     SavePrimitiveHelp(std::ostream &out, const c
har *hname, Option_t *option = "");
   static Bool_t    RecomputeAxisLimits(TAxis& destAxis, const T
Axis& anAxis);
   static Bool_t    SameLimitsAndNBins(const TAxis& axis1, const
 TAxis& axis2);

   virtual Double_t DoIntegral(Int_t ix1, Int_t ix2, Int_t iy1,
Int_t iy2, Int_t iz1, Int_t iz2, Double_t & err,
                               Option_t * opt, Bool_t doerr = kF
ALSE) const;

   virtual void     DoFillN(Int_t ntimes, const Double_t *x, con
st Double_t *w, Int_t stride=1);

   static bool CheckAxisLimits(const TAxis* a1, const TAxis* a2)
;
   static bool CheckBinLimits(const TAxis* a1, const TAxis* a2);
   static bool CheckBinLabels(const TAxis* a1, const TAxis* a2);
   static bool CheckEqualAxes(const TAxis* a1, const TAxis* a2);
```

```
   static bool CheckConsistentSubAxes(const TAxis *a1, Int_t fir
stBin1, Int_t lastBin1, const TAxis *a2, Int_t firstBin2=0, Int_
t lastBin2=0);
   static bool CheckConsistency(const TH1* h1, const TH1* h2);

public:
   // TH1 status bits
   enum {
      kNoStats     = BIT(9),  // don't draw stats box
      kUserContour = BIT(10), // user specified contour levels
      //kCanRebin    = BIT(11), // FIXME DEPRECATED - to be remo
ved, replaced by SetCanExtend / CanExtendAllAxes
      kLogX        = BIT(15), // X-axis in log scale
      kIsZoomed    = BIT(16), // bit set when zooming on Y axis
      kNoTitle     = BIT(17), // don't draw the histogram title
      kIsAverage   = BIT(18), // Bin contents are average (used
by Add)
      kIsNotW      = BIT(19)  // Histogram is forced to be not w
eighted even when the histogram is filled with weighted differen
t than 1.
   };
   // size of statistics data (size of  array used in GetStats()
/ PutStats )
   // s[0]  = sumw       s[1]  = sumw2
   // s[2]  = sumwx      s[3]  = sumwx2
   // s[4]  = sumwy      s[5]  = sumwy2   s[6]  = sumwxy
   // s[7]  = sumwz      s[8]  = sumwz2   s[9]  = sumwxz   s[10]
  = sumwyz
   // s[11] = sumwt      s[12] = sumwt2                 (11 and
12 used only by TProfile3D)
   enum {
      kNstat       = 13  // size of statistics data (up to TProf
ile3D)
   };


   virtual ~TH1();

   virtual Bool_t   Add(TF1 *h1, Double_t c1=1, Option_t *option=
"");
```

```
   virtual Bool_t   Add(const TH1 *h1, Double_t c1=1);//该函数是将
h1直方图进行c1倍的放大或缩小后加到当前直方图中。
   virtual Bool_t   Add(const TH1 *h, const TH1 *h2, Double_t c1=
1, Double_t c2=1); // *MENU* 将 h1 放大或缩小 c1 倍加上 h2 放大或缩小
 c2 倍相加赋给 h。
   virtual void     AddBinContent(Int_t bin);//在第 bin 个 bin 上
计数加一。
   virtual void     AddBinContent(Int_t bin, Double_t w);
   static  void     AddDirectory(Bool_t add=kTRUE);
   static  Bool_t   AddDirectoryStatus();
   virtual void     Browse(TBrowser *b);
   virtual Bool_t   CanExtendAllAxes() const;
   virtual Double_t Chi2Test(const TH1* h2, Option_t *option = "
UU", Double_t *res = 0) const;
   virtual Double_t Chi2TestX(const TH1* h2, Double_t &chi2, Int
_t &ndf, Int_t &igood,Option_t *option = "UU",  Double_t *res = 0
) const;
   virtual Double_t Chisquare(TF1 * f1, Option_t *option = "") c
onst;
   virtual void     ClearUnderflowAndOverflow();
   virtual Double_t ComputeIntegral(Bool_t onlyPositive = false)
;
   TObject*         Clone(const char* newname=0) const;//克隆直方
图，这里的 Close 是新开辟一块内存空间存储新直方图的，克隆之后，原始直方图的
修改或删除对克隆的直方图没有影响。而TH1D *hnew =h;这个只是将新建的对象指向
 h 的地址。
   virtual void     Copy(TObject &hnew) const;
   virtual void     DirectoryAutoAdd(TDirectory *);
   virtual Int_t    DistancetoPrimitive(Int_t px, Int_t py);
   virtual Bool_t   Divide(TF1 *f1, Double_t c1=1);
   virtual Bool_t   Divide(const TH1 *h1);
   virtual Bool_t   Divide(const TH1 *h1, const TH1 *h2, Double_
t c1=1, Double_t c2=1, Option_t *option=""); // *MENU*
   virtual void     Draw(Option_t *option="");//画图
   virtual TH1     *DrawCopy(Option_t *option="", const char * n
ame_postfix = "_copy") const;
   virtual TH1     *DrawNormalized(Option_t *option="", Double_t
 norm=1) const;
   virtual void     DrawPanel(); // *MENU*
   virtual Int_t    BufferEmpty(Int_t action=0);
```

```
   virtual void     Eval(TF1 *f1, Option_t *option="");
   virtual void     ExecuteEvent(Int_t event, Int_t px, Int_t py)
;
   virtual void     ExtendAxis(Double_t x, TAxis *axis);
   virtual TH1     *FFT(TH1* h_output, Option_t *option);
   virtual Int_t    Fill(Double_t x);//将数据填充进直方图
   virtual Int_t    Fill(Double_t x, Double_t w);
   virtual Int_t    Fill(const char *name, Double_t w);
   virtual void     FillN(Int_t ntimes, const Double_t *x, const
 Double_t *w, Int_t stride=1);
   virtual void     FillN(Int_t, const Double_t *, const Double_
t *, const Double_t *, Int_t) {;}
   virtual void     FillRandom(const char *fname, Int_t ntimes=5
000);
   virtual void     FillRandom(TH1 *h, Int_t ntimes=5000);
   virtual Int_t    FindBin(Double_t x, Double_t y=0, Double_t z=
0);//寻找x值对应的bin数 对一维直方图，返回 x 值所在 bin，若为二维直方图，
返回 （x，y） 所在 bin，三维同理。
   virtual Int_t    FindFixBin(Double_t x, Double_t y=0, Double_
t z=0) const;
   virtual Int_t    FindFirstBinAbove(Double_t threshold=0, Int_
t axis=1) const;
   virtual Int_t    FindLastBinAbove (Double_t threshold=0, Int_
t axis=1) const;
   virtual TObject *FindObject(const char *name) const;
   virtual TObject *FindObject(const TObject *obj) const;
   virtual TFitResultPtr    Fit(const char *formula ,Option_t *o
ption="" ,Option_t *goption="", Double_t xmin=0, Double_t xmax=0)
; // *MENU*
   virtual TFitResultPtr    Fit(TF1 *f1 ,Option_t *option="" ,Op
tion_t *goption="", Double_t xmin=0, Double_t xmax=0);
   virtual void     FitPanel(); // *MENU*
   TH1             *GetAsymmetry(TH1* h2, Double_t c2=1, Double_
t dc2=0);
   Int_t            GetBufferLength() const {return fBuffer ? (I
nt_t)fBuffer[0] : 0;}
   Int_t            GetBufferSize  () const {return fBufferSize;
}
   const   Double_t *GetBuffer() const {return fBuffer;}
   static  Int_t    GetDefaultBufferSize();
```

```
   virtual Double_t *GetIntegral();//ROOT 6  //TH1积分，当前bin值
为前面所有bin的累积，然后归一化（最大值为1），返回该数组
   TH1             *GetCumulative(Bool_t forward = kTRUE, const
char* suffix = "_cumulative") const;//ROOT 6 //TH1积分，当前bin值
为前面所有bin的累积。

   TList           *GetListOfFunctions() const { return fFunctio
ns; }

   virtual Int_t    GetNdivisions(Option_t *axis="X") const;
   virtual Color_t  GetAxisColor(Option_t *axis="X") const;
   virtual Color_t  GetLabelColor(Option_t *axis="X") const;
   virtual Style_t  GetLabelFont(Option_t *axis="X") const;
   virtual Float_t  GetLabelOffset(Option_t *axis="X") const;
   virtual Float_t  GetLabelSize(Option_t *axis="X") const;
   virtual Style_t  GetTitleFont(Option_t *axis="X") const;
   virtual Float_t  GetTitleOffset(Option_t *axis="X") const;
   virtual Float_t  GetTitleSize(Option_t *axis="X") const;
   virtual Float_t  GetTickLength(Option_t *axis="X") const;
   virtual Float_t  GetBarOffset() const {return Float_t(0.001*F
loat_t(fBarOffset));}
   virtual Float_t  GetBarWidth() const  {return Float_t(0.001*F
loat_t(fBarWidth));}
   virtual Int_t    GetContour(Double_t *levels=0);
   virtual Double_t GetContourLevel(Int_t level) const;
   virtual Double_t GetContourLevelPad(Int_t level) const;

   virtual Int_t    GetBin(Int_t binx, Int_t biny=0, Int_t binz=0
) const;
   virtual void     GetBinXYZ(Int_t binglobal, Int_t &binx, Int_
t &biny, Int_t &binz) const;
   virtual Double_t GetBinCenter(Int_t bin) const;//返回该bin的中
点坐标。
   virtual Double_t GetBinContent(Int_t bin) const;//返回该bin的计
数.
   virtual Double_t GetBinContent(Int_t bin, Int_t) const { retu
rn GetBinContent(bin); }
   virtual Double_t GetBinContent(Int_t bin, Int_t, Int_t) const
{ return GetBinContent(bin); }
   virtual Double_t GetBinError(Int_t bin) const;//Return value
```

```
of error associated to bin number bin. if the sum of squares of
weights has been defined (via Sumw2),this function returns the s
qrt(sum of w2). otherwise it returns the sqrt(contents) for this
 bin.
   virtual Double_t GetBinError(Int_t binx, Int_t biny) const {
return GetBinError(GetBin(binx, biny)); } // for 2D histograms o
nly
   virtual Double_t GetBinError(Int_t binx, Int_t biny, Int_t bi
nz) const { return GetBinError(GetBin(binx, biny, binz)); } // f
or 3D histograms only
   virtual Double_t GetBinErrorLow(Int_t bin) const;//Return low
er error associated to bin number bin. The error will depend on
the statistic option used will return the binContent - lower int
erval value
   virtual Double_t GetBinErrorUp(Int_t bin) const;//Return uppe
r error associated to bin number bin. The error will depend on t
he statistic option used will return the binContent - upper inte
rval value
   virtual EBinErrorOpt  GetBinErrorOption() const { return fBin
StatErrOpt; }
   virtual Double_t GetBinLowEdge(Int_t bin) const;//return bin
lower edge for 1D historam
   virtual Double_t GetBinWidth(Int_t bin) const;//返回bin的宽度。
   virtual Double_t GetBinWithContent(Double_t c, Int_t &binx, I
nt_t firstx=0, Int_t lastx=0,Double_t maxdiff=0) const;
   virtual void     GetCenter(Double_t *center) const;
   static  Bool_t   GetDefaultSumw2();
   TDirectory      *GetDirectory() const {return fDirectory;}
   virtual Double_t GetEntries() const;  //returns the number of
 entries
   virtual Double_t GetEffectiveEntries() const;
   virtual TF1     *GetFunction(const char *name) const;
   virtual Int_t    GetDimension() const { return fDimension; }
   virtual Double_t GetKurtosis(Int_t axis=1) const;//For axis =
1, 2 or 3 returns kurtosis of the histogram along x, y or z axis
.Kurtosis(gaussian(0, 1)) = 0. For axis =11, 12 or 13 returns th
e approximate standard error of kurtosis of the histogram along
x, y or z axis. Note, that since third and fourth moment are not
 calculated at the fill time, kurtosis and its standard error ar
e computed bin by bin
```

```
   virtual void     GetLowEdge(Double_t *edge) const;
   virtual Double_t GetMaximum(Double_t maxval=FLT_MAX) const;
   virtual Int_t    GetMaximumBin() const;
   virtual Int_t    GetMaximumBin(Int_t &locmax, Int_t &locmay,
Int_t &locmaz) const;
   virtual Double_t GetMaximumStored() const {return fMaximum;}
   virtual Double_t GetMinimum(Double_t minval=-FLT_MAX) const;
   virtual Int_t    GetMinimumBin() const;
   virtual Int_t    GetMinimumBin(Int_t &locmix, Int_t &locmiy,
Int_t &locmiz) const;
   virtual Double_t GetMinimumStored() const {return fMinimum;}
   virtual Double_t GetMean(Int_t axis=1) const;//returns the me
an value along axis. For axis = 1,2 or 3 returns the mean value
of the histogram along X,Y or Z axis. For axis = 11, 12, 13 retu
rns the standard error of the mean value of the histogram along
X, Y or Z axis. Note that the mean value/RMS is computed using t
he bins in the currently defined range (see TAxis::SetRange). By
 default the range includes all bins from 1 to nbins included, e
xcluding underflows and overflows. To force the underflows and o
verflows in the computation, one must call the static function T
H1::StatOverflows(kTRUE) before filling the histogram.
   virtual Double_t GetMeanError(Int_t axis=1) const;//Return st
andard error of mean of this histogram along the X axis.  Note t
hat the mean value/RMS is computed using the bins in the current
ly defined range (see TAxis::SetRange). By default the range inc
ludes  all bins from 1 to nbins included, excluding underflows a
nd overflows. To force the underflows and overflows in the compu
tation, one must call the static function TH1::StatOverflows(kTR
UE) before filling the histogram.  Also note, that although the
definition of standard error doesn't include the assumption of n
ormality, many uses of this feature implicitly assume it.
   virtual Int_t    GetNbinsX() const {return fXaxis.GetNbins();}
//获取该轴bin值
   virtual Int_t    GetNbinsY() const {return fYaxis.GetNbins();
}
   virtual Int_t    GetNbinsZ() const {return fZaxis.GetNbins();
}
   virtual Int_t    GetNcells() const {return fNcells; }//number
 of bins(1D), cells (2D) +U/Overflows
   virtual Double_t GetNormFactor() const {return fNormFactor;}/
```

```
/如果没有设置将会返回0。这个因子是将直方图面积设为该因子。设置该因子参考 Set
NormFactor(Double_t factor=1)。
   virtual char     *GetObjectInfo(Int_t px, Int_t py) const;
   Option_t         *GetOption() const {return fOption.Data();}

   TVirtualHistPainter *GetPainter(Option_t *option="");

   virtual Int_t    GetQuantiles(Int_t nprobSum, Double_t *q, co
nst Double_t *probSum=0);
   virtual Double_t GetRandom() const;//return a random number d
istributed according the histogram bin contents.  This function
checks if the bins integral exists. If not, the integral is eval
uated, normalized to one. The integral is automatically recomput
ed if the number of entries is not the same then when the integr
al was computed. NB Only valid for 1-d histograms. Use GetRandom
2 or 3 otherwise. If the histogram has a bin with negative conte
nt a NaN is returned
   virtual void     GetStats(Double_t *stats) const;
// fill the array stats from the contents of this histogram
// The array stats must be correctly dimensioned in the calling
program.
// stats[0] = sumw
// stats[1] = sumw2
// stats[2] = sumwx
// stats[3] = sumwx2
// If no axis-subrange is specified (via TAxis::SetRange), the a
rray stats is simply a copy of the statistics quantities compute
d at filling time.If a sub-range is specified, the function reco
mputes these quantities from the bin contents in the current axi
s range.
// Note that the mean value/RMS is computed using the bins in th
e currently defined range (see TAxis::SetRange). By default the
range includes all bins from 1 to nbins included, excluding unde
rflows and overflows. To force the underflows and overflows in t
he computation, one must call the static function TH1::StatOverf
lows(kTRUE) before filling the histogram.
   virtual Double_t GetStdDev(Int_t axis=1) const;//returns the
sigma distribution along axis
   virtual Double_t GetStdDevError(Int_t axis=1) const;
   virtual Double_t GetSumOfWeights() const;//Return the sum of
```

```
weights excluding under/overflows.
   virtual TArrayD *GetSumw2() {return &fSumw2;}
   virtual const TArrayD *GetSumw2() const {return &fSumw2;}
   virtual Int_t    GetSumw2N() const {return fSumw2.fN;}
           Double_t GetRMS(Int_t axis=1) const { return GetStdDe
v(axis); }
// For axis = 1,2 or 3 returns the Sigma value of the histogram
along X, Y or Z axis. For axis = 11, 12 or 13 returns the error
of RMS estimation along X, Y or Z axis for Normal distribution.
// Note that the mean value/sigma is computed using the bins in
the currently defined range (see TAxis::SetRange). By default th
e range includes all bins from 1 to nbins included, excluding un
derflows and overflows. To force the underflows and overflows in
 the computation, one must  call the static function TH1::StatOv
erflows(kTRUE) before filling  the histogram.
// Note that this function returns the Standard Deviation (Sigma
) of the distribution (not RMS). The Sigma estimate is computed
as Sqrt((1/N)*(Sum(x_i-x_mean)^2)). The name "RMS" was introduce
d many years ago (Hbook/PAW times).
           Double_t GetRMSError(Int_t axis=1) const { return Get
StdDevError(axis); }
// Return error of RMS estimation for Normal distribution.
// Note that the mean value/RMS is computed using the bins in th
e currently defined range (see TAxis::SetRange). By default the
range includes all bins from 1 to nbins included, excluding unde
rflows and overflows.  To force the underflows and overflows in
the computation, one must call the static function TH1::StatOver
flows(kTRUE) before filling the histogram.
// Value returned is standard deviation of sample standard devia
tion.
Note that it is an approximated value which is valid only in the
case that the original data distribution is Normal. The correct
one would require the 4-th momentum value, which cannot be accur
ately estimated from an histogram since the x-information for al
l entries is not kept.
   virtual Double_t GetSkewness(Int_t axis=1) const;
           TAxis*   GetXaxis()  { return &fXaxis; }//返回指向该坐
标轴 TAxis* 的指针，可对该轴进行设置。具体参考TAxis类的使用。
           TAxis*   GetYaxis()  { return &fYaxis; }//返回指向该坐
标轴 TAxis* 的指针，可对该轴进行设置。具体参考TAxis类的使用。
```

```
            TAxis*   GetZaxis()  { return &fZaxis; }//返回指向该坐
标轴 TAxis* 的指针，可对该轴进行设置。具体参考TAxis类的使用。
     const TAxis*   GetXaxis() const { return &fXaxis; }
     const TAxis*   GetYaxis() const { return &fYaxis; }
     const TAxis*   GetZaxis() const { return &fZaxis; }
   virtual Double_t Integral(Option_t *option="") const;//Return
 integral of bin contents. Only bins in the bins range are consi
dered. By default the integral is computed as the sum of bin con
tents in the range.  if option "width" is specified, the integra
l is the sum of the bin contents multiplied by the bin width in
x.
   virtual Double_t Integral(Int_t binx1, Int_t binx2, Option_t
*option="") const;//returns the integral of bin contents in a gi
ven bin range
   virtual Double_t IntegralAndError(Int_t binx1, Int_t binx2, D
ouble_t & err, Option_t *option="") const;
   virtual Double_t Interpolate(Double_t x);//Given a point x, a
pproximates the value via linear interpolation based on the two
nearest bin centers. 获取x的y值。其值为最近两个bin线性插值。
   virtual Double_t Interpolate(Double_t x, Double_t y);
   virtual Double_t Interpolate(Double_t x, Double_t y, Double_t
 z);
            Bool_t   IsBinOverflow(Int_t bin) const;//Return true
 if the bin is overflow.
            Bool_t   IsBinUnderflow(Int_t bin) const;//Return tru
e if the bin is overflow.
   virtual Double_t AndersonDarlingTest(const TH1 *h2, Option_t
*option="") const;
   virtual Double_t AndersonDarlingTest(const TH1 *h2, Double_t
&advalue) const;
   virtual Double_t KolmogorovTest(const TH1 *h2, Option_t *opti
on="") const;// statistical test of compatibility in shape betwe
en two histograms
   virtual void     LabelsDeflate(Option_t *axis="X");// Reduce
the number of bins for the axis passed in the option to the numb
er of bins having a label. The method will remove only the extra
 bins existing after the last "labeled" bin. Note that if there
are "un-labeled" bins present between "labeled" bins they will n
ot be removed.
   virtual void     LabelsInflate(Option_t *axis="X");//Double t
```

```
he number of bins for axis.Refill histogram. This function is ca
lled by TAxis::FindBin(const char *label).
   virtual void     LabelsOption(Option_t *option="h", Option_t
*axis="X");
   virtual Long64_t Merge(TCollection *list);
   virtual Bool_t   Multiply(TF1 *h1, Double_t c1=1);//Performs
the operation: this = this*c1*f1. if errors are defined (see TH1
::Sumw2), errors are also recalculated.  Only bins inside the fu
nction range are recomputed. IMPORTANT NOTE: If you intend to us
e the errors of this histogram later you should call Sumw2 befor
e making this operation. This is particularly important if you f
it the histogram after TH1::Multiply The function return kFALSE
if the Multiply operation failed.
   virtual Bool_t   Multiply(const TH1 *h1);// Multiply this his
togram by h1.  this = this*h1   If errors of this are available
(TH1::Sumw2), errors are recalculated.  Note that if h1 has Sumw
2 set, Sumw2 is automatically called for this if not already set
.   IMPORTANT NOTE: If you intend to use the errors of this hist
ogram later you should call Sumw2 before making this operation.
This is particularly important if you fit the histogram after TH
1::Multiply. The function return kFALSE if the Multiply operatio
n failed
   virtual Bool_t   Multiply(const TH1 *h1, const TH1 *h2, Doubl
e_t c1=1, Double_t c2=1, Option_t *option=""); // *MENU*  Replac
e contents of this histogram by multiplication of h1 by h2. this
 = (c1*h1)*(c2*h2)   If errors of this are available (TH1::Sumw2
), errors are recalculated. Note that if h1 or h2 have Sumw2 set
, Sumw2 is automatically called for this if not already set.   I
MPORTANT NOTE: If you intend to use the errors of this histogram
 later you should call Sumw2 before making this operation. This
is particularly important if you fit the histogram after TH1::Mu
ltiply. The function return kFALSE if the Multiply operation fai
led.
   virtual void     Paint(Option_t *option="");//Control routine
 to paint any kind of histograms.  This function is automaticall
y called by TCanvas::Update.(see TH1::Draw for the list of optio
ns)
   virtual void     Print(Option_t *option="") const;//Print som
e global quantities for this histogram. If option "base" is give
n, number of bins and ranges are also printed.  If option "range
```

```
" is given, bin contents and errors are also printed for all bin
s in the current range (default 1-->nbins).  If option "all" is
given, bin contents and errors are also printed for all bins inc
luding under and overflows.
   virtual void     PutStats(Double_t *stats);
   virtual TH1     *Rebin(Int_t ngroup=2, const char*newname="",
const Double_t *xbins=0);  // *MENU*
   virtual TH1     *RebinX(Int_t ngroup=2, const char*newname="")
 { return Rebin(ngroup,newname, (Double_t*) 0); }
   virtual void     Rebuild(Option_t *option="");
   virtual void     RecursiveRemove(TObject *obj);
   virtual void     Reset(Option_t *option="");//将其清除重置
// Reset this histogram: contents, errors, etc.
// if option "ICE" is specified, resets only Integral, Contents
and Errors.
// if option "ICES" is specified, resets only Integral, Contents
 , Errors and Statistics
// if option "M"   is specified, resets also Minimum and Maximum

// The option "ICE" is used when extending the histogram (in Ext
endAxis, LabelInflate, etc..)
// The option "ICES is used in combination with the buffer (see
BufferEmpty and BufferFill)
   virtual void     ResetStats();//Reset the statistics includin
g the number of entries and replace with values calculates from
bin content. The number of entries is set to the total bin conte
nt or (in case of weighted histogram) to number of effective ent
ries.
   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
   virtual void     Scale(Double_t c1=1, Option_t *option="");//
比例缩放
   virtual void     SetAxisColor(Color_t color=1, Option_t *axis=
"X");
   virtual void     SetAxisRange(Double_t xmin, Double_t xmax, O
ption_t *axis="X");
   virtual void     SetBarOffset(Float_t offset=0.25) {fBarOffse
t = Short_t(1000*offset);}
   virtual void     SetBarWidth(Float_t width=0.5) {fBarWidth =
Short_t(1000*width);}
```

```
   virtual void     SetBinContent(Int_t bin, Double_t content);
   virtual void     SetBinContent(Int_t bin, Int_t, Double_t con
tent) { SetBinContent(bin, content); }
   virtual void     SetBinContent(Int_t bin, Int_t, Int_t, Doubl
e_t content) { SetBinContent(bin, content); }
   virtual void     SetBinError(Int_t bin, Double_t error);
   virtual void     SetBinError(Int_t binx, Int_t biny, Double_t
 error);
   virtual void     SetBinError(Int_t binx, Int_t biny, Int_t bi
nz, Double_t error);
   virtual void     SetBins(Int_t nx, Double_t xmin, Double_t xm
ax);
   virtual void     SetBins(Int_t nx, const Double_t *xBins);
   virtual void     SetBins(Int_t nx, Double_t xmin, Double_t xm
ax, Int_t ny, Double_t ymin, Double_t ymax);
   virtual void     SetBins(Int_t nx, const Double_t *xBins, Int
_t ny, const Double_t *yBins);
   virtual void     SetBins(Int_t nx, Double_t xmin, Double_t xm
ax, Int_t ny, Double_t ymin, Double_t ymax,
                            Int_t nz, Double_t zmin, Double_t zm
ax);
   virtual void     SetBins(Int_t nx, const Double_t *xBins, Int
_t ny, const Double_t * yBins, Int_t nz,
                            const Double_t *zBins);
   virtual void     SetBinsLength(Int_t = -1) { } //redefined in
 derived classes
   virtual void     SetBinErrorOption(EBinErrorOpt type) { fBinS
tatErrOpt = type; }
   virtual void     SetBuffer(Int_t buffersize, Option_t *option=
"");
   virtual UInt_t   SetCanExtend(UInt_t extendBitMask);
   virtual void     SetContent(const Double_t *content);
   virtual void     SetContour(Int_t nlevels, const Double_t *le
vels=0);
   virtual void     SetContourLevel(Int_t level, Double_t value)
;
   static  void     SetDefaultBufferSize(Int_t buffersize=1000);
   static  void     SetDefaultSumw2(Bool_t sumw2=kTRUE);
   virtual void     SetDirectory(TDirectory *dir);
   virtual void     SetEntries(Double_t n) {fEntries = n;};
```

```
   virtual void     SetError(const Double_t *error);
   virtual void     SetLabelColor(Color_t color=1, Option_t *axi
s="X");
   virtual void     SetLabelFont(Style_t font=62, Option_t *axis=
"X");
   virtual void     SetLabelOffset(Float_t offset=0.005, Option_
t *axis="X");
   virtual void     SetLabelSize(Float_t size=0.02, Option_t *ax
is="X");

   /*
    * Set the minimum / maximum value for the Y axis (1-D histog
rams) or Z axis (2-D histograms)
    *   By default the maximum / minimum value used in drawing i
s the maximum / minimum value of the histogram
    * plus a margin of 10%. If these functions are called, the v
alues are used without any extra margin.
    */
   virtual void     SetMaximum(Double_t maximum = -1111) { fMaxi
mum = maximum; }; // *MENU*  Set the minimum / maximum value for
 the Y axis (1-D histograms) or Z axis (2-D histograms). By defa
ult the maximum / minimum value used in drawing is the maximum /
 minimum value of the histogram plus a margin of 10\%. If these
functions are called, the values are used without any extra marg
in.
   virtual void     SetMinimum(Double_t minimum = -1111) { fMini
mum = minimum; }; // *MENU*

   virtual void     SetName(const char *name); // *MENU*  Change
 the name of this histogram. Histograms are named objects in a T
HashList. We must update the hashlist if we change the name. We
protect this operation
   virtual void     SetNameTitle(const char *name, const char *t
itle);//Change the name and title of this histogram. Histograms
are named objects in a THashList. We must update the hashlist if
 we change the name
   virtual void     SetNdivisions(Int_t n=510, Option_t *axis="X"
);
   virtual void     SetNormFactor(Double_t factor=1) {fNormFacto
r = factor;}
```

```
   virtual void     SetStats(Bool_t stats=kTRUE); // *MENU*
   virtual void     SetOption(Option_t *option=" ") {fOption = o
ption;}
   virtual void     SetTickLength(Float_t length=0.02, Option_t
*axis="X");
   virtual void     SetTitleFont(Style_t font=62, Option_t *axis=
"X");
   virtual void     SetTitleOffset(Float_t offset=1, Option_t *a
xis="X");
   virtual void     SetTitleSize(Float_t size=0.02, Option_t *ax
is="X");
   virtual void     SetTitle(const char *title);  // *MENU*  Cha
nge (i.e. set) the title. if title is in the form "stringt;strin
gx;stringy;stringz" the histogram title is set to stringt, the x
 axis title to stringx, the y axis title to stringy, and the z a
xis title to stringz. To insert the character ";" in one of the
titles, one should use "\#;" or "\#semicolon".
   virtual void     SetXTitle(const char *title) {fXaxis.SetTitl
e(title);}
   virtual void     SetYTitle(const char *title) {fYaxis.SetTitl
e(title);}
   virtual void     SetZTitle(const char *title) {fZaxis.SetTitl
e(title);}
   virtual TH1     *ShowBackground(Int_t niter=20, Option_t *opt
ion="same"); // *MENU*
   virtual Int_t    ShowPeaks(Double_t sigma=2, Option_t *option=
"", Double_t threshold=0.05); // *MENU*
   virtual void     Smooth(Int_t ntimes=1, Option_t *option="");
// *MENU* Smooth bin contents of this histogram. if option conta
ins "R" smoothing is applied only to the bins defined in the X a
xis range (default is to smooth all bins). Bin contents are repl
aced by their smooth values. Errors (if any) are not modified. t
he smoothing procedure is repeated ntimes (default=1)
   static  void     SmoothArray(Int_t NN, Double_t *XX, Int_t nt
imes=1);//smooth array xx, translation of Hbook routine hsmoof.F
    based on algorithm 353QH twice presented by J. Friedman in P
roc.of the 1974 CERN School of Computing, Norway, 11-24 August,
1974.
   static  void     StatOverflows(Bool_t flag=kTRUE);
   virtual void     Sumw2(Bool_t flag = kTRUE);
```

```
   void              UseCurrentStyle();
   static  TH1      *TransformHisto(TVirtualFFT *fft, TH1* h_outp
ut,  Option_t *option);


   // TODO: Remove obsolete methods in v6-04
   virtual Double_t GetCellContent(Int_t binx, Int_t biny) const
                        { Obsolete("GetCellContent", "v6-00", "v
6-04"); return GetBinContent(GetBin(binx, biny)); }
   virtual Double_t GetCellError(Int_t binx, Int_t biny) const
                        { Obsolete("GetCellError", "v6-00", "v6-
04"); return GetBinError(binx, biny); }
   virtual void     RebinAxis(Double_t x, TAxis *axis)
                        { Obsolete("RebinAxis", "v6-00", "v6-04"
); ExtendAxis(x, axis); }
   virtual void     SetCellContent(Int_t binx, Int_t biny, Doubl
e_t content)
                        { Obsolete("SetCellContent", "v6-00", "v
6-04"); SetBinContent(GetBin(binx, biny), content); }
   virtual void     SetCellError(Int_t binx, Int_t biny, Double_
t content)
                        { Obsolete("SetCellError", "v6-00", "v6-
04"); SetBinError(binx, biny, content); }
```

code

```
TH1F *h1=new TH1F("stats name","title name",number of bins,min,m
ax);
TH1F *h1=new TH1F("stats name","title name;X axis;Y axis",number
 of bins,min,max);
 float x=gRandom->Uniform(-5.,5.);
 h1->Fill(x);                        //均匀分布
h1->FillRandom("gaus",50000);        //
h1->Draw();
```

```
//循环创建图片
TH1D  h[16];
char Stringtemp[256];
for(int i=0;i<16;i++){
  sprintf(Stringtemp,"h[%d]",i);//这样创建数组名不好，直接 "h%d"这样好

  h[i]=new TH1D(Stringtemp,"",4096,0,4096);
}
```

```
//默认情况下，Draw 将会清空当 前Pad 再画当前直方图，如想要一个图上多个直方图，将第二个及之后的直方图 Option_t 设为 "same"。
  TH1D *h=new TH1D("h","the title",60,-3.0,3.0);
  TH1D *h1=new TH1D("h1","",60,-3.0,3.0);
  TRandom *r =new TRandom();
  for (int i = 0; i<1000; ++i)
  {
    h->Fill(r->Gaus());
    h1->Fill(r->Gaus(1,1));
  }
  h->Draw();
  // h1->Draw();
  h1->Draw("same");
```

```
//Double_t   *GetIntegral();
//Return a pointer to the array of bins integral.
//返回一个指向数组的指针，当前值为 bin 从 0到当前 bin 面积所占的比例。
  int bin=60;
  TH1D *h=new TH1D("h","the title",bin,-3.0,3.0);
  TRandom *r =new TRandom();
  for (int i = 0; i<1000; ++i)
    {
      h->Fill(r->Gaus());
    }
  double *Integral;
  Integral=h->GetIntegral();
  for (int i = 0; i<bin; ++i)
    {
      std::cout<<i<<"  "<<Integral[i]<<std::endl;
    }
  h->Draw();
```

```
//克隆直方图，这里的 Close 是新开辟一块内存空间存储新直方图的，克隆之后，原始直方图的修改或删除对克隆的直方图没有影响。而TH1D *hnew =h;这个只是将新建的对象指向 h 的地址。
TH1D *h=new TH1D("h","the title",60,-3.0,3.0);
TRandom *r =new TRandom();
for (int i = 0; i<1000; ++i)
{
  h->Fill(r->Gaus());
}
TH1D *hnew = (TH1D*)h->Clone("hnew");
// TH1D *hnew =h;
h->Reset();
hnew->Draw();
```

example

TH2

继承 TH1

class

```
   virtual ~TH2();
   virtual Int_t    BufferEmpty(Int_t action=0);
   virtual void     Copy(TObject &hnew) const;
   virtual Int_t    Fill(Double_t x, Double_t y);
   virtual Int_t    Fill(Double_t x, Double_t y, Double_t w);
   virtual Int_t    Fill(Double_t x, const char *namey, Double_t
 w);
   virtual Int_t    Fill(const char *namex, Double_t y, Double_t
 w);
   virtual Int_t    Fill(const char *namex, const char *namey, D
ouble_t w);
   virtual void     FillN(Int_t, const Double_t *, const Double_
t *, Int_t) {;} //MayNotUse
   virtual void     FillN(Int_t ntimes, const Double_t *x, const
 Double_t *y, const Double_t *w, Int_t stride=1);
   virtual void     FillRandom(const char *fname, Int_t ntimes=5
000);
   virtual void     FillRandom(TH1 *h, Int_t ntimes=5000);
   virtual Int_t    FindFirstBinAbove(Double_t threshold=0, Int_
t axis=1) const;
   virtual Int_t    FindLastBinAbove (Double_t threshold=0, Int_
t axis=1) const;
   virtual void     FitSlicesX(TF1 *f1=0,Int_t firstybin=0, Int_
t lastybin=-1, Int_t cut=0, Option_t *option="QNR", TObjArray* a
rr = 0); // *MENU*
   virtual void     FitSlicesY(TF1 *f1=0,Int_t firstxbin=0, Int_
t lastxbin=-1, Int_t cut=0, Option_t *option="QNR", TObjArray* a
rr = 0); // *MENU*
   virtual Int_t    GetBin(Int_t binx, Int_t biny, Int_t binz = 0
) const;
   virtual Double_t GetBinWithContent2(Double_t c, Int_t &binx,
```

```
Int_t &biny, Int_t firstxbin=1, Int_t lastxbin=-1,Int_t firstybi
n=1, Int_t lastybin=-1, Double_t maxdiff=0) const;
   virtual Double_t GetBinContent(Int_t bin) const { return TH1:
:GetBinContent(bin); }
   virtual Double_t GetBinContent(Int_t binx, Int_t biny) const
{ return TH1::GetBinContent( GetBin(binx, biny) ); }
   virtual Double_t GetBinContent(Int_t binx, Int_t biny, Int_t)
const { return TH1::GetBinContent( GetBin(binx, biny) ); }
   using TH1::GetBinErrorLow;
   using TH1::GetBinErrorUp;
   virtual Double_t GetBinErrorLow(Int_t binx, Int_t biny) { ret
urn TH1::GetBinErrorLow( GetBin(binx, biny) ); }
   virtual Double_t GetBinErrorUp(Int_t binx, Int_t biny) { retu
rn TH1::GetBinErrorUp( GetBin(binx, biny) ); }
   virtual Double_t GetCorrelationFactor(Int_t axis1=1,Int_t axi
s2=2) const;
   virtual Double_t GetCovariance(Int_t axis1=1,Int_t axis2=2) c
onst;
   virtual void     GetRandom2(Double_t &x, Double_t &y);//返回满
足二维分布的x,y随机数
   virtual void     GetStats(Double_t *stats) const;
   virtual Double_t Integral(Option_t *option="") const;
   //virtual Double_t Integral(Int_t, Int_t, Option_t * ="") con
st {return 0;}
   using TH1::Integral;
   virtual Double_t Integral(Int_t binx1, Int_t binx2, Int_t bin
y1, Int_t biny2, Option_t *option="") const;
   virtual Double_t Integral(Int_t, Int_t, Int_t, Int_t, Int_t,
Int_t, Option_t * ="") const {return 0;}
   using TH1::IntegralAndError;
   virtual Double_t IntegralAndError(Int_t binx1, Int_t binx2, I
nt_t biny1, Int_t biny2, Double_t & err, Option_t *option="") co
nst;
   virtual Double_t Interpolate(Double_t x);
   virtual Double_t Interpolate(Double_t x, Double_t y);
   virtual Double_t Interpolate(Double_t x, Double_t y, Double_t
 z);
   virtual Double_t KolmogorovTest(const TH1 *h2, Option_t *opti
on="") const;
   virtual Long64_t Merge(TCollection *list);
```

```
   virtual TH2      *RebinX(Int_t ngroup=2, const char *newname="" );//合bin，几个合成一个
   virtual TH2      *RebinY(Int_t ngroup=2, const char *newname="" );//合bin，几个合成一个
   virtual TH2      *Rebin2D(Int_t nxgroup=2, Int_t nygroup=2, const char *newname="");//合bin，几个合成一个
      TProfile      *ProfileX(const char *name="_pfx", Int_t firstybin=1, Int_t lastybin=-1, Option_t *option="") const;   // *MENU*
      TProfile      *ProfileY(const char *name="_pfy", Int_t firstxbin=1, Int_t lastxbin=-1, Option_t *option="") const;   // *MENU*
         TH1D       *ProjectionX(const char *name="_px", Int_t firstybin=0, Int_t lastybin=-1, Option_t *option="") const; // *MENU*
         TH1D       *ProjectionY(const char *name="_py", Int_t firstxbin=0, Int_t lastxbin=-1, Option_t *option="") const; // *MENU*
   virtual void      PutStats(Double_t *stats);
   TH1D             *QuantilesX(Double_t prob = 0.5, const char * name = "_qx" ) const;
   TH1D             *QuantilesY(Double_t prob = 0.5, const char * name = "_qy" ) const;
   virtual void      Reset(Option_t *option="");
   virtual void      SetBinContent(Int_t bin, Double_t content);
   virtual void      SetBinContent(Int_t binx, Int_t biny, Double_t content) { SetBinContent(GetBin(binx, biny), content); }
   virtual void      SetBinContent(Int_t binx, Int_t biny, Int_t, Double_t content) { SetBinContent(GetBin(binx, biny), content); }
   virtual void      SetShowProjectionX(Int_t nbins=1);  // *MENU*

   virtual void      SetShowProjectionY(Int_t nbins=1);  // *MENU*

   virtual TH1      *ShowBackground(Int_t niter=20, Option_t *option="same");
   virtual Int_t     ShowPeaks(Double_t sigma=2, Option_t *option="", Double_t threshold=0.05); // *MENU*
   virtual void      Smooth(Int_t ntimes=1, Option_t *option="");  // *MENU*
```

code

```
TH2* h = new TH2D(/* name */ "h2",/* title */ "Hist with constan
t bin width",/* X-dimension */ 100, 0.0, 4.0,/* Y-dimension */ 2
00, -3.0, 1.5);
```

```
const Int_t XBINS = 5; const Int_t YBINS = 5;
Double_t xEdges[XBINS + 1] = {0.0, 0.2, 0.3, 0.6, 0.8, 1.0};
Double_t yEdges[YBINS + 1] = {-1.0, -0.4, -0.2, 0.5, 0.7, 1.0};
TH2* h = new TH2D("h2", "h2", XBINS, xEdges, YBINS, yEdges);
TAxis* xAxis = h->GetXaxis(); TAxis* yAxis = h->GetYaxis();
cout << "Third bin on Y-dimension: " << endl; // corresponds to
[-0.2, 0.5]
cout << "\tLower edge: " << yAxis->GetBinLowEdge(3) << endl;
cout << "\tCenter: " << yAxis->GetBinCenter(3) << endl;
cout << "\tUpper edge: " << yAxis->GetBinUpEdge(3) << endl;
```

example

```
//直方图的bin标签设置

const Int_t nx = 12;
const Int_t ny = 20;
char *month[nx]  = {"January","February","March","April","May","
June","July","August","September","October","November","December"
};
char *people[ny] = {"Jean","Pierre","Marie","Odile","Sebastien",
"Fons","Rene","Nicolas","Xavier","Greg","Bjarne","Anton","Otto",
"Eddy","Peter","Pasha","Philippe","Suzanne","Jeff","Valery"};
TCanvas *c1 = new TCanvas("c1","demo bin labels",10,10,800,800);
c1->SetGrid();
c1->SetLeftMargin(0.15);
c1->SetBottomMargin(0.15);
TH2F *h = new TH2F("h","test",nx,0,nx,ny,0,ny);
for (Int_t i=0;i<5000;i++) {
  h->Fill(gRandom->Gaus(0.5*nx,0.2*nx), gRandom->Gaus(0.5*ny,0.2
*ny));
 }
h->SetStats(0);
for (i=1;i<=nx;i++) h->GetXaxis()->SetBinLabel(i,month[i-1]);//
直方图每个bin设置标签名字。
for (i=1;i<=ny;i++) h->GetYaxis()->SetBinLabel(i,people[i-1]);//
一维二维一样的添加方式。
h->Draw("text");
```

TH2Poly

TH3

TLatex

继承 TText, TAttLine

To draw Mathematical Formula.

TLatex's purpose is to write mathematical equations. The syntax is very similar to the Latex's one.

When the font precision (see TAttText) is low (0 or 1), TLatex is painted as a normal TText, the control characters are not interpreted.

Subscripts and superscripts are made with the `_` and `^` commands. These commands can be combined to make complicated subscript and superscript expressions. You may adjust the display of subscripts and superscripts by using the two functions SetIndiceSize(Double_t), which set relative size of subscripts and superscripts, and SetLimitIndiceSize(Int_t), which set limits for text resizing of subscripts and superscripts.

```
TLatex can display dozens of special mathematical symbols. A few
 of them, such
as + and > , are produced by typing the corresponding
keyboard character.

TLatex provides 4 kinds of proportional delimiters:

    #[]{....} or "a la" Latex #left[.....#right] : big square br
ackets
    #{}{....} or              #left{.....#right} : big curly bra
ckets
    #||{....} or              #left|.....#right| : big absolute
value symbols
    #(){....} or              #left(.....#right) : big parenthes
es


The command to produce a lowercase Greek letter is obtained by a
dding a
# to the name of the letter. For an uppercase Greek letter, just
capitalize the first letter of the command name. Some letters ha
ve two
representations. The name of the second one (the "variation") st
arts with "var".

Several kind of accents are available:

   #hat    = Begin_Latex #hat{a} End_Latex
   #check  = Begin_Latex #check{a} End_Latex
   #acute  = Begin_Latex #acute{a} End_Latex
   #grave  = Begin_Latex #grave{a} End_Latex
   #dot    = Begin_Latex #dot{a} End_Latex
   #ddot   = Begin_Latex #ddot{a} End_Latex
   #tilde  = Begin_Latex #tilde{a} End_Latex
```

The class TMathText is a TeX math formulae interpreter. It uses plain TeX syntax and uses "\" as control instead of "#". If a piece of text containing "\" is given to TLatex then TMathText is automatically invoked. Therefore, as histograms' titles,

axis titles, labels etc ... are drawn using TLatex, the TMathText syntax can be used for them also.

class

```
      TLatex();
      TLatex(Double_t x, Double_t y, const char *text);
      TLatex(const TLatex &text);
      virtual ~TLatex();
      void             Copy(TObject &text) const;

      TLatex          *DrawLatex(Double_t x, Double_t y, const c
har *text);
/// Make a copy of this object with the new parameters
/// And copy object attributes

      TLatex          *DrawLatexNDC(Double_t x, Double_t y, const
 char *text);/// Draw this TLatex with new coordinates in NDC.

      Double_t         GetHeight() const;/// Return height of cu
rrent pad in pixels
      Double_t         GetXsize();/// Return size of the formula
 along X in pad coordinates
      Double_t         GetYsize();/// Return size of the formula
 along Y in pad coordinates
      void             GetBoundingBox(UInt_t &w, UInt_t &h, Bool
_t angle = kFALSE);/// Return text size in pixels
      virtual void     Paint(Option_t *option="");/// Paint.
      virtual void     PaintLatex(Double_t x, Double_t y, Double
_t angle, Double_t size, const char *text);
/// Main drawing function
/// Warning: Unlike most others "XYZ::PaintXYZ" methods, PaintLa
tex modifies
///          the TLatex data members.

      virtual void     SavePrimitive(std::ostream &out, Option_t
 *option = "");/// Save primitive as a C++ statement(s) on outpu
t stream out
      virtual void     SetIndiceSize(Double_t factorSize);/// Se
t relative size of subscripts and superscripts
      virtual void     SetLimitIndiceSize(Int_t limitFactorSize);
/// Set limit for text resizing of subscripts and superscripts
```

code

```
   TCanvas *cl = new TCanvas("cl","cl",10,10,700,500);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Double_t dy = 1./7.;
   Tl.DrawText(.1, dy,   "x^{2y} :");    Tl.DrawLatex(.5, dy,
"x^{2y}");
   Tl.DrawText(.1, 2*dy, "x_{2y} :");    Tl.DrawLatex(.5, 2*dy,
"x_{2y}");
   Tl.DrawText(.1, 3*dy, "x^{y^{2}} :"); Tl.DrawLatex(.5, 3*dy,
"x^{y^{2}}");
   Tl.DrawText(.1, 4*dy, "x^{y_{1}} :"); Tl.DrawLatex(.5, 4*dy,
"x^{y_{1}}");
   Tl.DrawText(.1, 5*dy, "x^{y}_{1} :"); Tl.DrawLatex(.5, 5*dy,
"x^{y}_{1}");
   Tl.DrawText(.1, 6*dy, "x_{1}^{y} :"); Tl.DrawLatex(.5, 6*dy,
"x_{1}^{y}");
```

上下角标

```
// The best way to put the subscripts and superscripts before th
e character and not
// after, is to use an empty character:
   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "{}^{40}_{20}Ca :"); Tl.DrawLatex(.5, .5
, "{}^{40}_{20}Ca");
```

```
   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "x = #frac{y+z/2}{y^{2}+1} :"); Tl.Draw
Latex(.5, .5, "x = #frac{y+z/2}{y^{2}+1}");
```

```
// The #sqrt command produces the square root of its argument; i
t has
// an optional first argument for other roots.

   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "#sqrt{10} #sqrt[3]{10} :"); Tl.DrawLat
ex(.5, .5, "#sqrt{10} #sqrt[3]{10}");
```

```
// One can change the font, the text color, or the text size at
any time using :  #font[font-number]{...}, #color[color-number]{
...} and #scale[scale-factor]{...}
   TCanvas *cl = new TCanvas("cl","cl",10,10,900,300);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Double_t dy = 1./4.;
   Tl.DrawText(.01, dy,   "#font[12]{Times Italic} and #font[22]
{Times bold} :");    Tl.DrawLatex(.7, dy,   "#font[12]{Times Ita
lic} and #font[22]{Times bold}");
   Tl.DrawText(.01, 2*dy, "#color[2]{Red} and #color[4]{Blue} :"
);    Tl.DrawLatex(.7, 2*dy, "#color[2]{Red} and #color[4]{Blue}"
);
   Tl.DrawText(.01, 3*dy, "#scale[1.2]{Bigger} and #scale[0.8]{S
maller} :"); Tl.DrawLatex(.7, 3*dy, "#scale[1.2]{Bigger} and #sc
ale[0.8]{Smaller}");
```

```
// Text can be split in two lines via the command #splitline.

   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "#splitline{21 April 2003}{14:02:30} :"
); Tl.DrawLatex(.6, .5, "#splitline{21 April 2003}{14:02:30}");
```

```
// The special sign: #slash draws a slash on top of the text bet
ween brackets:

   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "#slash{E}_{T} :"); Tl.DrawLatex(.5, .5
, "#slash{E}_{T}");
```

```
// Bar and vectors sign are done the following way:

   TCanvas *cl = new TCanvas("cl","cl",10,10,700,100);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Tl.DrawText(.1, .5,   "#bar{a} and #vec{a} :"); Tl.DrawLatex(
.5, .5, "#bar{a} and #vec{a}");
```

```
// One can change the font, the text color, or the text size at
any time using :
// #font[font-number]{...}, #color[color-number]{...}
// and #scale[scale-factor]{...}

   TCanvas *cl = new TCanvas("cl","cl",10,10,900,300);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Double_t dy = 1./4.;
   Tl.DrawText(.01, dy,   "#font[12]{Times Italic} and #font[22]
{Times bold} :");    Tl.DrawLatex(.7, dy,   "#font[12]{Times Ita
lic} and #font[22]{Times bold}");
   Tl.DrawText(.01, 2*dy, "#color[2]{Red} and #color[4]{Blue} :"
);    Tl.DrawLatex(.7, 2*dy, "#color[2]{Red} and #color[4]{Blue}"
);
   Tl.DrawText(.01, 3*dy, "#scale[1.2]{Bigger} and #scale[0.8]{S
maller} :"); Tl.DrawLatex(.7, 3*dy, "#scale[1.2]{Bigger} and #sc
ale[0.8]{Smaller}");
```

```
// The two commands #kern and #lower enable a better control
// over character placement. The command #kern[(Float_t)dx]{text
} moves
// the output string horizontally by the fraction dx of its leng
th.
// Similarly, #lower[(Float_t)dy]{text} shifts the text up or do
wn by
// the fraction dy of its height.

   TCanvas *cl = new TCanvas("cl","cl",10,10,900,300);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   TLatex Tt; Tt.SetTextFont(43); Tt.SetTextSize(12);
   Double_t dy = 1./7.;
   Tl.DrawLatex(.5,   dy, "Positive k#kern[0.3]{e}#kern[0.3]{r}#
kern[0.3]{n}#kern[0.3]{i}#kern[0.3]{n}#kern[0.3]{g}");
   Tt.DrawText(.01, 2*dy, "Positive k#kern[0.3]{e}#kern[0.3]{r}#
kern[0.3]{n}#kern[0.3]{i}#kern[0.3]{n}#kern[0.3]{g} :");
   Tl.DrawLatex(.5, 3*dy, "Negative k#kern[-0.3]{e}#kern[-0.3]{r
}#kern[-0.3]{n}#kern[-0.3]{i}#kern[-0.3]{n}#kern[-0.3]{g}");
   Tt.DrawText(.01, 4*dy, "Negative k#kern[-0.3]{e}#kern[-0.3]{r
}#kern[-0.3]{n}#kern[-0.3]{i}#kern[-0.3]{n}#kern[-0.3]{g} :");
   Tl.DrawLatex(.5, 5*dy, "Vertical a#lower[0.2]{d}#lower[0.4]{j
}#lower[0.1]{u}#lower[-0.1]{s}#lower[-0.3]{t}#lower[-0.4]{m}#low
er[-0.2]{e}#lower[0.1]{n}t");
   Tt.DrawText(.01, 6*dy, "Vertical a#lower[0.2]{d}#lower[0.4]{j
}#lower[0.1]{u}#lower[-0.1]{s}#lower[-0.3]{t}#lower[-0.4]{m}#low
er[-0.2]{e}#lower[0.1]{n}t :");
```

```
// Text can be turned italic or boldface using the commands #it
and #bf.

   TCanvas *cl = new TCanvas("cl","cl",10,10,900,300);
   TLatex Tl; Tl.SetTextFont(43); Tl.SetTextSize(20);
   Double_t dy = 1./3.;
   Tl.DrawText(.01, dy,   "abc#alpha#beta#gamma, #it{abc#alpha#b
eta#gamma} :");    Tl.DrawLatex(.7, dy,   "abc#alpha#beta#gamma,
 #it{abc#alpha#beta#gamma}");
   Tl.DrawText(.01, 2*dy, "#bf{bold}, #it{italic}, #bf{#it{bold
italic}}, #bf{#bf{unbold}}} :");    Tl.DrawLatex(.7, 2*dy, "#bf{
bold}, #it{italic}, #bf{#it{bold italic}}, #bf{#bf{unbold}}}");
```

example

```
// The TText alignment rules apply to the TLatex objects with on
e exception
// concerning the vertical alignment:

// - if the vertical alignment = 1 , subscripts are not taken in
to account
// - if the vertical alignment = 0 , the text is aligned to the
box surrounding
                                        the full text with sub and
superscripts
// This is illustrated by the following example:

   TCanvas Tlva("Tlva","Tlva",500,500);
   Tlva.SetGrid();
   Tlva.DrawFrame(0,0,1,1);
   const char *longstring = "K_{S}... K^{*0}... #frac{2s}{#pi#al
pha^{2}} #frac{d#sigma}{dcos#theta} (e^{+}e^{-} #rightarrow f#ba
r{f} ) = #left| #frac{1}{1 - #Delta#alpha} #right|^{2} (1+cos^{2
}#theta)";

   TLatex latex;
   latex.SetTextSize(0.025);
   latex.SetTextAlign(13);  //align at top
```

```
    latex.DrawLatex(.2,.9,"K_{S}");
    latex.DrawLatex(.3,.9,"K^{*0}");
    latex.DrawLatex(.2,.8,longstring);

    latex.SetTextAlign(12);  //centered
    latex.DrawLatex(.2,.6,"K_{S}");
    latex.DrawLatex(.3,.6,"K^{*0}");
    latex.DrawLatex(.2,.5,longstring);

    latex.SetTextAlign(11);  //default bottom alignment
    latex.DrawLatex(.2,.4,"K_{S}");
    latex.DrawLatex(.3,.4,"K^{*0}");
    latex.DrawLatex(.2,.3,longstring);

    latex.SetTextAlign(10);  //special bottom alignment
    latex.DrawLatex(.2,.2,"K_{S}");
    latex.DrawLatex(.3,.2,"K^{*0}");
    latex.DrawLatex(.2,.1,longstring);

    latex.SetTextAlign(12);
    latex.SetTextFont(72);
    latex.DrawLatex(.1,.80,"13");
    latex.DrawLatex(.1,.55,"12");
    latex.DrawLatex(.1,.35,"11");
    latex.DrawLatex(.1,.18,"10");
    return Tlva;
```

```
   TCanvas ex1("ex1","Latex",500,600);
   TLatex Tl;
   Tl.SetTextAlign(12);
   Tl.SetTextSize(0.04);
   Tl.DrawLatex(0.1,0.8,"1)   C(x) = d #sqrt{#frac{2}{#lambdaD}}
  #int^{x}_{0}cos(#frac{#pi}{2}t^{2})dt");
   Tl.DrawLatex(0.1,0.6,"2)   C(x) = d #sqrt{#frac{2}{#lambdaD}}
  #int^{x}cos(#frac{#pi}{2}t^{2})dt");
   Tl.DrawLatex(0.1,0.4,"3)   R = |A|^{2} = #frac{1}{2}(#[]{#fra
c{1}{2}+C(V)}^{2}+#[]{#frac{1}{2}+S(V)}^{2})");
   Tl.DrawLatex(0.1,0.2,"4)   F(t) = #sum_{i=-#infty}^{#infty}A(
i)cos#[]{#frac{i}{t+i}}");
   return ex1;
```

```
   TCanvas ex2("ex2","Latex",500,300);
   TLatex Tl;
   Tl.SetTextAlign(23);
   Tl.SetTextSize(0.08);
   Tl.DrawLatex(0.5,0.95,"e^{+}e^{-}#rightarrowZ^{0}#rightarrowI
#bar{I}, q#bar{q}");
   Tl.DrawLatex(0.5,0.75,"|#vec{a}#bullet#vec{b}|=#Sigmaa^{i}_{j
k}+b^{bj}_{i}");
   Tl.DrawLatex(0.5,0.5,"i(#partial_{#mu}#bar{#psi}#gamma^{#mu}+
m#bar{#psi}=0#Leftrightarrow(#Box+m^{2})#psi=0");
   Tl.DrawLatex(0.5,0.3,"L_{em}=eJ^{#mu}_{em}A_{#mu} , J^{#mu}_{
em}=#bar{I}#gamma_{#mu}I , M^{j}_{i}=#SigmaA_{#alpha}#tau^{#alph
aj}_{i}");
   return ex2;
```

```
   TCanvas ex3("ex3","Latex",500,300);
   TPaveText pt(.1,.1,.9,.9);
   pt.AddText("#frac{2s}{#pi#alpha^{2}}  #frac{d#sigma}{dcos#the
ta} (e^{+}e^{-} #rightarrow f#bar{f} ) = ");
   pt.AddText("#left| #frac{1}{1 - #Delta#alpha} #right|^{2} (1+
cos^{2}#theta");
   pt.AddText("+ 4 Re #left{ #frac{2}{1 - #Delta#alpha} #chi(s)
#[]{#hat{g}_{#nu}^{e}#hat{g}_{#nu}^{f} (1 + cos^{2}#theta) + 2 #
hat{g}_{a}^{e}#hat{g}_{a}^{f} cos#theta) } #right}");
   pt.SetLabel("Born equation");
   pt.Draw();
   return ex3;
```

TLegend

继承 TPave , TAttText

This class displays a legend box (TPaveText) containing several legend entries.

Each legend entry is made of a reference to a ROOT object, a text label and an option specifying which graphical attributes (marker/line/fill) should be displayed.

The legend contains a histogram, a function and a graph. The histogram is put in the legend using its reference pointer whereas the graph and the function are added using their names. Note that, because TGraph constructors do not have the TGraph name as parameter, the graph name should be specified using the SetName method.

TLegend inherits from TAttText therefore changing any text attributes (text alignment, font, color...) on a legend will changed the text attributes on each line.

Note that the TPad class has a method to build automatically a legend for all objects in the pad. It is called TPad::BuildLegend().

Each item in the legend is added using the AddEntry method. This method defines the object to be added (by reference or name), the label associated to this object and an option which a combination of:

- L: draw line associated with TAttLine if obj inherits from TAttLine
- P: draw polymarker associated with TAttMarker if obj inherits from TAttMarker
- F: draw a box with fill associated wit TAttFill if obj inherits TAttFill
- E: draw vertical error bar

class

```
   TLegend();
   TLegend( Double_t x1, Double_t y1, Double_t x2, Double_t y2,
            const char* header = "", Option_t* option="brNDC" );
/// Normal constructor.
/// A TLegend is a Pave with several TLegendEntry(s).
```

```
/// x1,y1,x2,y2 are the coordinates of the Legend in the current
 pad
/// (in normalised coordinates by default)
/// "header" is the title that will be displayed at the top of t
he legend
/// it is treated like a regular entry and supports TLatex. The
default
/// is no header (header = 0).
/// The options are the same as for TPave Default = "brNDC"

   virtual ~TLegend();
   TLegend( const TLegend &legend );

   TLegendEntry   *AddEntry(const TObject* obj, const char* labe
l = "", Option_t* option = "lpf" );
/// Add a new entry to this legend. "obj" is the object to be re
presented.
/// "label" is the text you wish to associate with obj in the le
gend.
/// If "label" is null or empty, the title of the object will be
 used.
/// Options are:
///  - L: draw line associated with TAttLine if obj inherits fro
m TAttLine
///  - P: draw polymarker associated with TAttMarker if obj inhe
rits from TAttMarker
///  - F: draw a box with fill associated wit TAttFill if obj in
herits TAttFill
///  - E: draw vertical error bar if option "L" is also specified


   TLegendEntry   *AddEntry(const char *name, const char* label
= "", Option_t* option = "lpf" );
/// Add a new entry to this legend. "name" is the name of an obj
ect in the pad to
/// be represented label is the text you wish to associate with
obj in the legend
/// if label is null or empty, the title of the object will be u
sed.
/// Options are:
```

```
///  - L: draw line associated with TAttLine if obj inherits fro
m TAttLine
///  - P: draw polymarker associated with TAttMarker if obj inhe
rits from TAttMarker
///  - F: draw a box with fill associated wit TAttFill if obj in
herits TAttFill
///  - E: draw vertical error bar if option "L" is also specified


   virtual void    Clear( Option_t* option = "" ); // *MENU* ///
 Clear all entries in this legend, including the header.
   virtual void    Copy( TObject &obj ) const; /// Copy this leg
end into "obj".
   virtual void    DeleteEntry(); // *MENU* /// Delete entry at
the mouse position.
   virtual void    Draw( Option_t* option = "" );/// Draw this l
egend with its current attributes.
   virtual void    EditEntryAttFill();/// Edit the fill attribut
es for the entry pointed by the mouse.
   virtual void    EditEntryAttLine();/// Edit the line attribut
es for the entry pointed by the mouse.
   virtual void    EditEntryAttMarker();/// Edit the marker attr
ibutes for the entry pointed by the mouse.
   virtual void    EditEntryAttText();/// Edit the text attribut
es for the entry pointed by the mouse.
   Float_t         GetColumnSeparation() const { return fColumnS
eparation; }
   TLegendEntry   *GetEntry() const;
/// Get entry pointed to by the mouse.
/// This method is mostly a tool for other methods inside this c
lass.

   Float_t         GetEntrySeparation() const { return fEntrySep
aration; }
   virtual const char *GetHeader() const;
/// Returns the header, which is the title that appears at the t
op
/// of the legend.

   TList          *GetListOfPrimitives() const {return fPrimitiv
```

```
es;}
   Float_t         GetMargin() const { return fMargin; }
   Int_t           GetNColumns() const { return fNColumns; }
   Int_t           GetNRows() const;/// Get the number of rows.
   virtual void    InsertEntry( const char* objectName = "",const
 char* label = "",
                              Option_t* option = "lpf" ); // *M
ENU*
/// Add a new entry before the entry at the mouse position.

   virtual void    Paint( Option_t* option = "" );/// Paint this
 legend with its current attributes.
   virtual void    PaintPrimitives();/// Paint the entries (list
 of primitives) for this legend.
   virtual void    Print( Option_t* option = "" ) const;/// Pain
t this legend with its current attributes.

/// Dump this TLegend and its contents.

   virtual void    RecursiveRemove(TObject *obj);
/// Reset the legend entries pointing to "obj".

   virtual void    SavePrimitive(std::ostream &out, Option_t *op
tion  = "");
/// Save this legend as C++ statements on output stream out
/// to be used with the SaveAs .C option.

   void            SetDefaults() { fEntrySeparation = 0.1f; fMar
gin = 0.25f; fNColumns = 1; fColumnSeparation = 0.0f; }
   void            SetColumnSeparation( Float_t columnSeparation
 )
                     { fColumnSeparation = columnSeparation; } /
/ *MENU*
   virtual void    SetEntryLabel( const char* label ); // *MENU*
/// Edit the label of the entry pointed to by the mouse.

   virtual void    SetEntryOption( Option_t* option ); // *MENU*
/// Edit the option of the entry pointed to by the mouse.

   void            SetEntrySeparation( Float_t entryseparation )
```

```
                        { fEntrySeparation = entryseparation; } //
*MENU*
   virtual void    SetHeader( const char *header = "" );  // *ME
NU*
/// Sets the header, which is the "title" that appears at the to
p of the legend.

   void            SetMargin( Float_t margin ) { fMargin = margi
n; } // *MENU*
   void            SetNColumns( Int_t nColumns ); // *MENU*
/// Set the number of columns for the legend. The header, if set
, is given
/// its own row. After that, every nColumns entries are inserted
 into the
/// same row. For example, if one calls legend.SetNColumns(2), a
nd there
/// is no header, then the first two TObjects added to the legen
d will be
/// in the first row, the next two will appear in the second row
, and so on.
```

code

```
// In particular it can be interesting to change the text aligne
ment that way. In
// order to have a base-line vertical alignment instead of a cen
tered one simply do:

   leg->SetTextAlign(13);

// or

   leg->SetTextAlign(11);
```

```
The default value of some TLegend attributes can be changed using

gStyle. The default settings are:

   SetLegendBorderSize(1);
   SetLegendFillColor(0);
   SetLegendFont(42);
   SetLegendTextSize(0.);

// The global attributes change the default values for the next
created legends.

// Text attributes can be also changed individually on each lege
nd entry:

   TLegendEntry *le = leg->AddEntry(h1,"Histogram filled with ra
ndom numbers","f");
   le->SetTextColor(kBlue);;
```

```
// When an object is added by name, a scan is performed on the l
ist of objects
// contained in the current pad (gPad) and also in the possible
TMultiGraph and
// THStack present in the pad. If a matching name is found, the
corresponding
// object is added in the legend using its pointer.

   TCanvas *c1 = new TCanvas("c1","c1",600,500);
   gStyle->SetOptStat(0);

   TH1F *h1 = new TH1F("h1","TLegend Example",200,-10,10);
   h1->FillRandom("gaus",30000);
   h1->SetFillColor(kGreen);
   h1->SetFillStyle(3003);
   h1->Draw();

   TF1 *f1=new TF1("f1","1000*TMath::Abs(sin(x)/x)",-10,10);
   f1->SetLineColor(kBlue);
```

```
    f1->SetLineWidth(4);
    f1->Draw("same");

    const Int_t n = 20;
    Double_t x[n], y[n], ex[n], ey[n];
    for (Int_t i=0;i<n;i++) {
       x[i]  = i*0.1;
       y[i]  = 1000*sin(x[i]+0.2);
       x[i]  = 17.8*x[i]-8.9;
       ex[i] = 1.0;
       ey[i] = 10.*i;
    }
    TGraphErrors *gr = new TGraphErrors(n,x,y,ex,ey);
    gr->SetName("gr");
    gr->SetLineColor(kRed);
    gr->SetLineWidth(2);
    gr->SetMarkerStyle(21);
    gr->SetMarkerSize(1.3);
    gr->SetMarkerColor(7);
    gr->Draw("P");

    leg = new TLegend(0.1,0.7,0.48,0.9);
    leg->SetHeader("The Legend Title");
    leg->AddEntry(h1,"Histogram filled with random numbers","f");
    leg->AddEntry("f1","Function abs(#frac{sin(x)}{x})","l");
    leg->AddEntry("gr","Graph with error bars","lep");
    leg->Draw();

    return c1;
```

```
It is possible to draw the legend entries over several columns u
sing
the method SetNColumns() like in the following example.

   TCanvas *c3 = new TCanvas("c2","c2",500,300);

   TLegend* leg = new TLegend(0.2, 0.2, .8, .8);
   TH1* h = new TH1F("", "", 1, 0, 1);

   leg-> SetNColumns(2);

   leg->AddEntry(h, "Column 1 line 1", "l");
   leg->AddEntry(h, "Column 2 line 1", "l");
   leg->AddEntry(h, "Column 1 line 2", "l");
   leg->AddEntry(h, "Column 2 line 2", "l");

   leg->Draw();
   return c3;
```

```
TLegend *legend = new TLegend(0.55,0.65,0.76,0.82);
legend->SetHeader("The Legend Title");
legend->SetTextSize(0.05);
legend->SetBorderSize(0);
legend->AddEntry(h1,"All nations","");
legend->AddEntry(h2,"French only","");
legend->Draw();
```

example

TLegendEntry

继承 TObject, TAttText, TAttLine, TAttFill, TAttMarker

Storage class for one entry of a TLegend

class

```
   TLegendEntry();
   TLegendEntry(const TObject *obj, const char *label = 0, Optio
n_t *option="lpf" );
/// TLegendEntry normal constructor for one entry in a TLegend.
/// obj is the object this entry will represent. If obj has
/// line/fill/marker attributes, then the TLegendEntry will disp
lay
/// these attributes.
/// label is the text that will describe the entry, it is displa
yed using
/// TLatex, so may have a complex format.
/// option may have values
///  - L draw line associated w/ TAttLine if obj inherits from T
AttLine
///  - P draw polymarker assoc. w/ TAttMarker if obj inherits fr
om TAttMarker
///  - F draw a box with fill associated w/ TAttFill if obj inhe
rits TAttFill
///    default is object = "LPF"

   TLegendEntry( const TLegendEntry &entry );
   virtual ~TLegendEntry();
   virtual void          Copy( TObject &obj ) const;/// copy thi
s TLegendEntry into obj
   virtual const char   *GetLabel() const { return fLabel.Data()
; }
   virtual TObject      *GetObject() const { return fObject; }
   virtual Option_t     *GetOption() const { return fOption.Data
(); }
```

```
   virtual void          Print( Option_t *option = "" ) const;//
/ dump this TLegendEntry to std::cout
   virtual void          SaveEntry( std::ostream &out, const char
 *name );
/// Save this TLegendEntry as C++ statements on output stream out

///  to be used with the SaveAs .C option

   virtual void          SetLabel( const char *label = "" ) { fL
abel = label; } // *MENU*
   virtual void          SetObject(TObject* obj );/// (re)set th
e obj pointed to by this entry
   virtual void          SetObject( const char *objectName );  /
/ *MENU* /// (re)set the obj pointed to by this entry
   virtual void          SetOption( Option_t *option="lpf" ) { f
Option = option; } // *MENU*
```

code

example

TLine

继承 TObject, TAttLine, TAttBBox2D

class

```
   // TLine status bits
   enum {
      kLineNDC    = BIT(14), // Use NDC coordinates
      kVertical   = BIT(15), // Line is vertical
      kHorizontal = BIT(16)  // Line is horizontal
   };

   TLine();/// Line default constructor.
   TLine(Double_t x1, Double_t y1,Double_t x2, Double_t  y2);///
 Line normal constructor.
   TLine(const TLine &line);/// Line copy constructor.
   virtual ~TLine();

   void                 Copy(TObject &line) const;/// Copy this
line to line.
   virtual Int_t        DistancetoPrimitive(Int_t px, Int_t py);
/// Compute distance from point px,py to a line.
   virtual TLine       *DrawLine(Double_t x1, Double_t y1,Double
_t x2, Double_t y2);/// Draw this line with new coordinates.
   virtual TLine       *DrawLineNDC(Double_t x1, Double_t y1,Dou
ble_t x2, Double_t y2);/// Draw this line with new coordinates i
n NDC.
   virtual void         ExecuteEvent(Int_t event, Int_t px, Int_
t py);
/// Execute action corresponding to one event.
///  This member function is called when a line is clicked with
the locator
///  If Left button clicked on one of the line end points, this
point
///     follows the cursor until button is released.
///  if Middle button clicked, the line is moved parallel to its
```

```
elf
///     until the button is released.

   Double_t             GetX1() const {return fX1;}
   Double_t             GetX2() const {return fX2;}
   Double_t             GetY1() const {return fY1;}
   Double_t             GetY2() const {return fY2;}
   Bool_t               IsHorizontal();/// Check whether this li
ne is to be drawn horizontally.
   Bool_t               IsVertical();/// Check whether this line
 is to be drawn vertically.
   virtual void         ls(Option_t *option="") const;/// List t
his line with its attributes.
   virtual void         Paint(Option_t *option="");/// Paint thi
s line with its current attributes.
   virtual void         PaintLine(Double_t x1, Double_t y1,Doubl
e_t x2, Double_t  y2);/// Draw this line with new coordinates.
   virtual void         PaintLineNDC(Double_t u1, Double_t v1,Do
uble_t u2, Double_t  v2);/// Draw this line with new coordinates
 in NDC.
   virtual void         Print(Option_t *option="") const;
   virtual void         SavePrimitive(std::ostream &out, Option_
t *option = "");/// Save primitive as a C++ statement(s) on outp
ut stream out
   virtual void         SetNDC(Bool_t isNDC=kTRUE);/// Set NDC m
ode on if isNDC = kTRUE, off otherwise
   void                 SetHorizontal(Bool_t set = kTRUE); // *T
OGGLE* *GETTER=IsHorizontal
/// Force the line to be drawn horizontally.
/// Makes fY2 equal to fY1. The line length is kept.
/// TArrow and TGaxis also get this function by inheritance.

   void                 SetVertical(Bool_t set = kTRUE); // *TOG
GLE* *GETTER=IsVertical
/// Force the line to be drawn vertically.
/// Makes fX2 equal to fX1. The line length is kept.
/// TArrow and TGaxis also get this function by inheritance.

   virtual void         SetX1(Double_t x1) {fX1=x1;}
   virtual void         SetX2(Double_t x2) {fX2=x2;}
```

```
   virtual void         SetY1(Double_t y1) {fY1=y1;}
   virtual void         SetY2(Double_t y2) {fY2=y2;}
   virtual Rectangle_t  GetBBox();/// Return the bounding Box of
 the Line
   virtual TPoint       GetBBoxCenter();/// Return the center of
 the BoundingBox as TPoint in pixels
   virtual void         SetBBoxCenter(const TPoint &p);/// Set c
enter of the BoundingBox
   virtual void         SetBBoxCenterX(const Int_t x);/// Set X
coordinate of the center of the BoundingBox
   virtual void         SetBBoxCenterY(const Int_t y);/// Set Y
coordinate of the center of the BoundingBox
   virtual void         SetBBoxX1(const Int_t x);
/// Set left hand side of BoundingBox to a value
/// (resize in x direction on left)

   virtual void         SetBBoxX2(const Int_t x);
/// Set right hand side of BoundingBox to a value
/// (resize in x direction on right)

   virtual void         SetBBoxY1(const Int_t y);
/// Set top of BoundingBox to a value (resize in y direction on
top)

   virtual void         SetBBoxY2(const Int_t y);
/// Set bottom of BoundingBox to a value
/// (resize in y direction on bottom)
```

code

example

TLinearFitter

TList

继承 TSeqCollection

A doubly linked list. All classes inheriting from TObject can be inserted in a TList.

All classes inheriting from TObject can be inserted in a TList. Before being inserted into the list the object pointer is wrapped in a TObjLink object which contains, besides the object pointer also a previous and next pointer.

class

```
   typedef TListIter Iterator_t;

   TList() : fFirst(0), fLast(0), fCache(0), fAscending(kTRUE) {
 }
   TList(TObject *) : fFirst(0), fLast(0), fCache(0), fAscending
(kTRUE) { } // for backward compatibility, don't use
   virtual           ~TList();
   virtual void      Clear(Option_t *option="");
/// Remove all objects from the list. Does not delete the objects

/// unless the TList is the owner (set via SetOwner()) and option

/// "nodelete" is not set.
/// If option="nodelete" then don't delete any heap objects that
 were
/// marked with the kCanDelete bit, otherwise these objects will
 be
/// deleted (this option is used by THashTable::Clear()).

   virtual void      Delete(Option_t *option="");
/// Remove all objects from the list AND delete all heap based o
bjects.
/// If option="slow" then keep list consistent during delete. Th
is allows
/// recursive list operations during the delete (e.g. during the
```

```
 dtor
/// of an object in this list one can still access the list to s
earch for
/// other not yet deleted objects).

   virtual TObject  *FindObject(const char *name) const;
/// Find an object in this list using its name. Requires a seque
ntial
/// scan till the object has been found. Returns 0 if object wit
h specified
/// name is not found. This method overrides the generic FindObj
ect()
/// of TCollection for efficiency reasons.

   virtual TObject  *FindObject(const TObject *obj) const;
/// Find an object in this list using the object's IsEqual()
/// member function. Requires a sequential scan till the object
has
/// been found. Returns 0 if object is not found.
/// This method overrides the generic FindObject() of TCollectio
n for
/// efficiency reasons.

   virtual TIterator *MakeIterator(Bool_t dir = kIterForward) co
nst;

   virtual void      Add(TObject *obj) { AddLast(obj); }
   virtual void      Add(TObject *obj, Option_t *opt) { AddLast(
obj, opt); }
   virtual void      AddFirst(TObject *obj);/// Add object at th
e beginning of the list.
   virtual void      AddFirst(TObject *obj, Option_t *opt);
/// Add object at the beginning of the list and also store optio
n.
/// Storing an option is useful when one wants to change the beh
aviour
/// of an object a little without having to create a complete new

/// copy of the object. This feature is used, for example, by th
e Draw()
```

```
/// method. It allows the same object to be drawn in different w
ays.

   virtual void      AddLast(TObject *obj);/// Add object at the
 end of the list.
   virtual void      AddLast(TObject *obj, Option_t *opt);
/// Add object at the end of the list and also store option.
/// Storing an option is useful when one wants to change the beh
aviour
/// of an object a little without having to create a complete new

/// copy of the object. This feature is used, for example, by th
e Draw()
/// method. It allows the same object to be drawn in different w
ays.

   virtual void      AddAt(TObject *obj, Int_t idx);/// Insert o
bject at position idx in the list.
   virtual void      AddAfter(const TObject *after, TObject *obj)
;/// Insert object after object after in the list.
   virtual void      AddAfter(TObjLink *after, TObject *obj);
/// Insert object after the specified ObjLink object. If after =
 0 then add
/// to the tail of the list. An ObjLink can be obtained by loopi
ng over a list
/// using the above describe iterator method 3.

   virtual void      AddBefore(const TObject *before, TObject *o
bj);/// Insert object before object before in the list.
   virtual void      AddBefore(TObjLink *before, TObject *obj);
/// Insert object before the specified ObjLink object. If before
 = 0 then add
/// to the head of the list. An ObjLink can be obtained by loopi
ng over a list
/// using the above describe iterator method 3.

   virtual TObject  *Remove(TObject *obj);/// Remove object from
 the list.
   virtual TObject  *Remove(TObjLink *lnk);/// Remove object lin
k (and therefore the object it contains) from the list.
```

```
   virtual void      RemoveLast();/// Remove the last object of
the list.
   virtual void      RecursiveRemove(TObject *obj);
/// Remove object from this collection and recursively remove th
e object
/// from all other objects (and collections).

   virtual TObject  *At(Int_t idx) const;/// Returns the object
at position idx. Returns 0 if idx is out of range.
   virtual TObject  *After(const TObject *obj) const;
/// Returns the object after object obj. Obj is found using the
/// object's IsEqual() method.  Returns 0 if obj is last in list.


   virtual TObject  *Before(const TObject *obj) const;
/// Returns the object before object obj. Obj is found using the
/// object's IsEqual() method.  Returns 0 if obj is first in lis
t.

   virtual TObject  *First() const;/// Return the first object i
n the list. Returns 0 when list is empty.
   virtual TObjLink *FirstLink() const { return fFirst; }
   virtual TObject **GetObjectRef(const TObject *obj) const;///
Return address of pointer to obj
   virtual TObject  *Last() const;/// Return the last object in
the list. Returns 0 when list is empty.
   virtual TObjLink *LastLink() const { return fLast; }

   virtual void      Sort(Bool_t order = kSortAscending);
/// Sort linked list. Real sorting is done in private function D
oSort().
/// The list can only be sorted when is contains objects of a so
rtable
/// class.

   Bool_t            IsAscending() { return fAscending; }
```

code

```
// There are basically four ways to iterate over a TList (in ord
er
// of preference, if not forced by other constraints):

//   1. Using the R__FOR_EACH macro:

GetListOfPrimitives()->R__FOR_EACH(TObject,Paint)(option);

//  2. Using the TList iterator TListIter (via the wrapper class
 TIter):

TIter next(GetListOfPrimitives());
while ((TObject *obj = next()))
   obj->Draw(next.GetOption());

//  3. Using the TList iterator TListIter and std::for_each algo
rithm:

  // A function object, which will be applied to each element
  // of the given range.
struct STestFunctor {
   bool operator()(TObject *aObj) {
      ...
      return true;
   }
}
...
...
TIter iter(mylist);
for_each( iter.Begin(), TIter::End(), STestFunctor() );

//  4. Using the TObjLink list entries (that wrap the TObject*):

TObjLink *lnk = GetListOfPrimitives()->FirstLink();
while (lnk) {
   lnk->GetObject()->Draw(lnk->GetOption());
   lnk = lnk->Next();
}
```

```
//  5. Using the TList's After() and Before() member functions:

TFree *idcur = this;
while (idcur) {
   ...
   ...
   idcur = (TFree*)GetListOfFree()->After(idcur);
         }

// Methods 2, 3 and 4 can also easily iterate backwards using ei
ther
// a backward TIter (using argument kIterBackward) or by using
// LastLink() and lnk->Prev() or by using the Before() member.
```

example

TMath

TMathBase

TMatrixT

TMatrixTBase

TMatrixTSparse

TMatrixTSym

TMemFile

TMinuit

TMinuit2TraceObject

TMinuitMinimizer

TMLPAnalyzer

TMonitor

TMultiGraph

继承 TNamed

class

```
   TMultiGraph();
   TMultiGraph(const char *name, const char *title);
   virtual ~TMultiGraph();

   virtual void      Add(TGraph *graph, Option_t *chopt="");
/// Add a new graph to the list of graphs.
/// Note that the graph is now owned by the TMultigraph.
/// Deleting the TMultiGraph object will automatically delete th
e graphs.
/// You should not delete the graphs when the TMultigraph is sti
ll active.

   virtual void      Add(TMultiGraph *multigraph, Option_t *chop
t="");
/// Add all the graphs in "multigraph" to the list of graphs.
/// If "chopt" is defined all the graphs in "multigraph" will be
 added with
/// the "chopt" option.
/// If "chopt" is undefined each graph will be added with the op
tion it had
/// in "multigraph".

   virtual void      Browse(TBrowser *b);/// Browse multigraph.
   virtual Int_t     DistancetoPrimitive(Int_t px, Int_t py);
/// Compute distance from point px,py to each graph.

   virtual void      Draw(Option_t *chopt="");
/// Draw this multigraph with its current attributes.
///   Options to draw a graph are described in TGraphPainter.
///  The drawing option for each TGraph may be specified as an o
ptional
```

```
///  second argument of the Add function. You can use GetGraphDr
awOption
///  to return this option.
///  If a draw option is specified, it will be used to draw the
graph,
///  otherwise the graph will be drawn with the option specified
 in
///  TMultiGraph::Draw. Use GetDrawOption to return the option s
pecified
///  when drawing the TMultiGraph.

   virtual TFitResultPtr Fit(const char *formula ,Option_t *opti
on="" ,Option_t *goption="", Axis_t xmin=0, Axis_t xmax=0);
/// Fit this graph with function with name fname.
///  interface to TF1::Fit(TF1 *f1...

   virtual TFitResultPtr Fit(TF1 *f1 ,Option_t *option="" ,Optio
n_t *goption="", Axis_t rxmin=0, Axis_t rxmax=0);
/// Fit this multigraph with function f1.
///   In this function all graphs of the multigraph are fitted s
imultaneously
///   f1 is an already predefined function created by TF1.
///   Predefined functions such as gaus, expo and poln are autom
atically
///   created by ROOT.
///
///   The list of fit options is given in parameter option.
///      option = "W"  Set all errors to 1
///             = "U" Use a User specified fitting algorithm (vi
a SetFCN)
///             = "Q" Quiet mode (minimum printing)
///             = "V" Verbose mode (default is between Q and V)
///             = "B" Use this option when you want to fix one o
r more parameters
///                   and the fitting function is like "gaus","e
xpo","poln","landau".
///             = "R" Use the Range specified in the function ra
nge
///             = "N" Do not store the graphics function, do not
 draw
```

```
///             = "0" Do not plot the result of the fit. By defa
ult the fitted function
///                   is drawn unless the option"N" above is spe
cified.
///             = "+" Add this new fitted function to the list o
f fitted functions
///                   (by default, any previous function is dele
ted)
///             = "C" In case of linear fitting, not calculate t
he chisquare
///                    (saves time)
///             = "F" If fitting a polN, switch to minuit fitter
///             = "ROB" In case of linear fitting, compute the L
TS regression
///                     coefficients (robust(resistant) regressi
on), using
///                     the default fraction of good points
///               "ROB=0.x" - compute the LTS regression coeffic
ients, using
///                           0.x as a fraction of good points
///
///   When the fit is drawn (by default), the parameter goption
may be used
///   to specify a list of graphics options. See TGraph::Paint f
or a complete
///   list of these options.
///
///   In order to use the Range option, one must first create a
function
///   with the expression to be fitted. For example, if your gra
ph
///   has a defined range between -4 and 4 and you want to fit a
 gaussian
///   only in the interval 1 to 3, you can do:
///        TF1 *f1 = new TF1("f1","gaus",1,3);
///        graph->Fit("f1","R");
///
///   who is calling this function
///   ============================
///   Note that this function is called when calling TGraphError
```

```
s::Fit
///   or TGraphAsymmErrors::Fit ot TGraphBentErrors::Fit
///   see the discussion below on the errors calulation.
///
///   Setting initial conditions
///   ==========================
///   Parameters must be initialized before invoking the Fit fun
ction.
///   The setting of the parameter initial values is automatic f
or the
///   predefined functions : poln, expo, gaus, landau. One can h
owever disable
///   this automatic computation by specifying the option "B".
///   You can specify boundary limits for some or all parameters
 via
///        f1->SetParLimits(p_number, parmin, parmax);
///   if parmin>=parmax, the parameter is fixed
///   Note that you are not forced to fix the limits for all par
ameters.
///   For example, if you fit a function with 6 parameters, you
can do:
///     func->SetParameters(0,3.1,1.e-6,0.1,-8,100);
///     func->SetParLimits(4,-10,-4);
///     func->SetParLimits(5, 1,1);
///   With this setup, parameters 0->3 can vary freely
///   Parameter 4 has boundaries [-10,-4] with initial value -8
///   Parameter 5 is fixed to 100.
///
///  Fit range
///  =========
///  The fit range can be specified in two ways:
///    - specify rxmax > rxmin (default is rxmin=rxmax=0)
///    - specify the option "R". In this case, the function will
 be taken
///      instead of the full graph range.
///
///  Changing the fitting function
///  =============================
///   By default a chi2 fitting function is used for fitting the
 TGraphs's.
```

```
///   The function is implemented in FitUtil::EvaluateChi2.
///   In case of TGraphErrors an effective chi2 is used
///   (see TGraphErrors fit in TGraph::Fit) and is implemented in

///   FitUtil::EvaluateChi2Effective
///   To specify a User defined fitting function, specify option
 "U" and
///   call the following functions:
///     TVirtualFitter::Fitter(mygraph)->SetFCN(MyFittingFunctio
n)
///   where MyFittingFunction is of type:
///   extern void MyFittingFunction(Int_t &npar, Double_t *gin,
Double_t &f, Double_t *u, Int_t flag);
///
///  Access to the fit result
///  ========================
///  The function returns a TFitResultPtr which can hold a  poin
ter to a TFitResult object.
///  By default the TFitResultPtr contains only the status of th
e fit and it converts
///  automatically to an integer. If the option "S" is instead u
sed, TFitResultPtr contains
///  the TFitResult and behaves as a smart pointer to it. For ex
ample one can do:
///     TFitResultPtr r = graph->Fit("myFunc","S");
///     TMatrixDSym cov = r->GetCovarianceMatrix();  //  to acce
ss the covariance matrix
///     Double_t par0   = r->Parameter(0); // retrieve the value
 for the parameter 0
///     Double_t err0   = r->ParError(0); // retrieve the error
for the parameter 0
///     r->Print("V");     // print full information of fit incl
uding covariance matrix
///     r->Write();        // store the result in a file
///
///   The fit parameters, error and chi2 (but not covariance mat
rix) can be retrieved also
///   from the fitted function.
///
///
```

```
///   Associated functions
///   ====================
///  One or more object (typically a TF1*) can be added to the l
ist
///  of functions (fFunctions) associated to each graph.
///  When TGraph::Fit is invoked, the fitted function is added t
o this list.
///  Given a graph gr, one can retrieve an associated function
///  with:  TF1 *myfunc = gr->GetFunction("myfunc");
///
///  If the graph is made persistent, the list of
///  associated functions is also persistent. Given a pointer (s
ee above)
///  to an associated function myfunc, one can retrieve the func
tion/fit
///  parameters with calls such as:
///    Double_t chi2 = myfunc->GetChisquare();
///    Double_t par0 = myfunc->GetParameter(0); //value of 1st p
arameter
///    Double_t err0 = myfunc->GetParError(0);  //error on first
 parameter
///
///   Fit Statistics
///   ==============
///  You can change the statistics box to display the fit parame
ters with
///  the TStyle::SetOptFit(mode) method. This mode has four digi
ts.
///  mode = pcev  (default = 0111)
///    v = 1;  print name/values of parameters
///    e = 1;  print errors (if e=1, v must be 1)
///    c = 1;  print Chisquare/Number of degress of freedom
///    p = 1;  print Probability
///
///  For example: gStyle->SetOptFit(1011);
///  prints the fit probability, parameter names/values, and err
ors.
///  You can change the position of the statistics box with thes
e lines
///  (where g is a pointer to the TGraph):
```

```
///
///  Root > TPaveStats *st = (TPaveStats*)g->GetListOfFunctions(
)->FindObject("stats")
///  Root > st->SetX1NDC(newx1); //new x start position
///  Root > st->SetX2NDC(newx2); //new x end position

   virtual void      FitPanel(); // *MENU*
/// Display a panel with all histogram fit options.
/// See class TFitPanel for example

   virtual Option_t *GetGraphDrawOption(const TGraph *gr) const;
/// Return the draw option for the TGraph gr in this TMultiGraph.

/// The return option is the one specified when calling TMultiGr
aph::Add(gr,option).

   virtual void      LeastSquareLinearFit(Int_t ndata, Double_t
&a0, Double_t &a1, Int_t &ifail, Double_t xmin, Double_t xmax);
/// Least square linear fit without weights.
///  Fit a straight line (a0 + a1*x) to the data in this graph.
///  ndata:  number of points to fit
///  first:  first point number to fit
///  last:   last point to fit O(ndata should be last-first
///  ifail:  return parameter indicating the status of the fit (
ifail=0, fit is OK)
///   extracted from CERNLIB LLSQ: Translated to C++ by Rene Brun


   virtual void      LeastSquareFit(Int_t m, Double_t *a, Double
_t xmin, Double_t xmax);
/// Least squares lpolynomial fitting without weights.
///  m     number of parameters
///  a     array of parameters
///  first 1st point number to fit (default =0)
///  last  last point number to fit (default=fNpoints-1)
///   based on CERNLIB routine LSQ: Translated to C++ by Rene Br
un

   virtual void      InitPolynom(Double_t xmin, Double_t xmax);
/// Compute Initial values of parameters for a polynom.
```

```
   virtual void      InitExpo(Double_t xmin, Double_t xmax);
/// Compute Initial values of parameters for an exponential.

   virtual void      InitGaus(Double_t xmin, Double_t xmax);
/// Compute Initial values of parameters for a gaussian.

   virtual Int_t     IsInside(Double_t x, Double_t y) const;
/// Return 1 if the point (x,y) is inside one of the graphs 0 ot
herwise.

   TH1F              *GetHistogram() const;
/// Returns a pointer to the histogram used to draw the axis.
/// Takes into account the two following cases.
///    1- option 'A' was specified in TMultiGraph::Draw. Return
fHistogram
///    2- user had called TPad::DrawFrame. return pointer to hfr
ame histogram

   TF1               *GetFunction(const char *name) const;
/// Return pointer to function with name.
/// Functions such as TGraph::Fit store the fitted function in t
he list of
/// functions of this graph.

   TList             *GetListOfGraphs() const { return fGraphs; }
   TList             *GetListOfFunctions();  // non const method
(create list if empty)
/// Return pointer to list of functions.
/// If pointer is null create the list

   const TList       *GetListOfFunctions() const { return fFuncti
ons; }
   TAxis             *GetXaxis() const;
/// Get x axis of the graph.
/// This method returns a valid axis only after the TMultigraph
has been drawn.

   TAxis             *GetYaxis() const;
/// Get y axis of the graph.
```

```
/// This method returns a valid axis only after the TMultigraph
has been drawn.

   virtual void      Paint(Option_t *chopt="");/// Paint all the
 graphs of this multigraph.
   void              PaintPads(Option_t *chopt="");/// Divides t
he active pad and draws all Graphs in the Multigraph separately.
   void              PaintPolyLine3D(Option_t *chopt="");/// Pai
nt all the graphs of this multigraph as 3D lines.
   virtual void      Print(Option_t *chopt="") const;/// Print t
he list of graphs.
   virtual void      RecursiveRemove(TObject *obj);
/// Recursively remove this object from a list. Typically implem
ented
/// by classes that can contain multiple references to a same ob
ject.

   virtual void      SavePrimitive(std::ostream &out, Option_t *
option = "");/// Save primitive as a C++ statement(s) on output
stream out.
   virtual void      SetMaximum(Double_t maximum=-1111); /// Set
 multigraph maximum.
   virtual void      SetMinimum(Double_t minimum=-1111); /// Set
 multigraph minimum.
```

code

```
// A TMultiGraph is a collection of TGraph (or derived) objects.
 It allows to
// manipulate a set of graphs as a single entity. In particular,
 when drawn,
// the X and Y axis ranges are automatically computed such as al
l the graphs
// will be visible.

TGraph *gr1 = new TGraph(...
TGraphErrors *gr2 = new TGraphErrors(...
TMultiGraph *mg = new TMultiGraph();
mg->Add(gr1,"lp");
mg->Add(gr2,"cp");
mg->Draw("a");
```

```
// The number of graphs in a multigraph can be retrieve with:

mg->GetListOfGraphs()->GetSize();
```

```
// The axis titles can be modified the following way:

TMultiGraph *mg = new TMultiGraph;
mg->SetTitle("title;xaxis title; yaxis title");
mg->Add(g1);
mg->Add(g2);
mg->Draw("apl");
```

example

```
//Draw three graphs with an exclusion zone.一个画板上画出多个Graph。


TCanvas *c1 = new TCanvas("c1","Exclusion graphs examples",200,10
,600,400);
c1->SetGrid();
```

```
TMultiGraph *mg = new TMultiGraph();
mg->SetTitle("Exclusion graphs");

const Int_t n = 35;
Double_t x1[n], x2[n], x3[n], y1[n], y2[n], y3[n];
for (Int_t i=0;i<n;i++) {
  x1[i]  = i*0.1;
  x2[i]  = x1[i];
  x3[i]  = x1[i]+.5;
  y1[i] = 10*sin(x1[i]);
  y2[i] = 10*cos(x1[i]);
  y3[i] = 10*sin(x1[i])-2;
 }

TGraph *gr1 = new TGraph(n,x1,y1);
gr1->SetLineColor(2);
gr1->SetLineWidth(1504);
gr1->SetFillStyle(3005);

TGraph *gr2 = new TGraph(n,x2,y2);
gr2->SetLineColor(4);
gr2->SetLineWidth(-2002);
gr2->SetFillStyle(3004);
gr2->SetFillColor(9);

TGraph *gr3 = new TGraph(n,x3,y3);
gr3->SetLineColor(5);
gr3->SetLineWidth(-802);
gr3->SetFillStyle(3002);
gr3->SetFillColor(2);

mg->Add(gr1);
mg->Add(gr2);
mg->Add(gr3);
mg->Draw("AC");
```

```
   c0 = new TCanvas("c1","multigraph L3",200,10,700,500);
   c0->SetFrameFillColor(30);

   TMultiGraph *mg = new TMultiGraph();

   TGraph *gr1 = new TGraph(); gr1->SetLineColor(kBlue);
   TGraph *gr2 = new TGraph(); gr2->SetLineColor(kRed);
   TGraph *gr3 = new TGraph(); gr3->SetLineColor(kGreen);
   TGraph *gr4 = new TGraph(); gr4->SetLineColor(kOrange);

   Double_t dx = 6.28/100;
   Double_t x  = -3.14;

   for (int i=0; i<=100; i++) {
      x = x+dx;
      gr1->SetPoint(i,x,2.*TMath::Sin(x));
      gr2->SetPoint(i,x,TMath::Cos(x));
      gr3->SetPoint(i,x,TMath::Cos(x*x));
      gr4->SetPoint(i,x,TMath::Cos(x*x*x));
   }

   mg->Add(gr4); gr4->SetTitle("Cos(x*x*x)"); gr4->SetLineWidth(3
);
   mg->Add(gr3); gr3->SetTitle("Cos(x*x)")  ; gr3->SetLineWidth(3
);
   mg->Add(gr2); gr2->SetTitle("Cos(x)")    ; gr2->SetLineWidth(3
);
   mg->Add(gr1); gr1->SetTitle("2*Sin(x)")  ; gr1->SetLineWidth(3
);

   mg->Draw("a fb l3d");
   return c0;
```

```
TCanvas *c1 = new TCanvas("c1","c1",600,400);

Double_t px1[2] = {2.,4.};
Double_t dx1[2] = {0.1,0.1};
Double_t py1[2] = {2.1,4.0};
Double_t dy1[2] = {0.3,0.2};

Double_t px2[2] = {3.,5.};
Double_t dx2[2] = {0.1,0.1};
Double_t py2[2] = {3.2,4.8};
Double_t dy2[2] = {0.3,0.2};

gStyle->SetOptFit(0001);

TGraphErrors *g1 = new TGraphErrors(2,px1,py1,dx1,dy1);
g1->SetMarkerStyle(21);
g1->SetMarkerColor(2);

TGraphErrors *g2 = new TGraphErrors(2,px2,py2,dx2,dy2);
g2->SetMarkerStyle(22);
g2->SetMarkerColor(3);

TMultiGraph *g = new TMultiGraph();
g->Add(g1);
g->Add(g2);

g->Draw("AP");

g->Fit("pol1","FQ");
return c1;
```

```
// The axis limits can be changed the like for TGraph. The same
methods apply on the multigraph.
// Note the two differents ways to change limits on X and Y axis.


   TCanvas *c2 = new TCanvas("c2","c2",600,400);

   TGraph *g[3];
   Double_t x[10] = {0,1,2,3,4,5,6,7,8,9};
   Double_t y[10] = {1,2,3,4,5,5,4,3,2,1};
   TMultiGraph *mg = new TMultiGraph();
   for (int i=0; i<3; i++) {
      g[i] = new TGraph(10, x, y);
      g[i]->SetMarkerStyle(20);
      g[i]->SetMarkerColor(i+2);
      for (int j=0; j<10; j++) y[j] = y[j]-1;
      mg->Add(g[i]);
   }
   mg->Draw("APL");
   mg->GetXaxis()->SetTitle("E_{#gamma} (GeV)");
   mg->GetYaxis()->SetTitle("Coefficients");

   // Change the axis limits
   gPad->Modified();
   mg->GetXaxis()->SetLimits(1.5,7.5);
   mg->SetMinimum(0.);
   mg->SetMaximum(10.);
```

```
// The method TPad::BuildLegend is able to extract the graphs in
side a
// multigraph. The following example demonstrate this.

   TCanvas *c3 = new TCanvas("c3","c3",600, 400);

   TMultiGraph * mg = new TMultiGraph("mg","mg");

   const Int_t size = 10;
```

```
    double px[size];
    double py1[size];
    double py2[size];
    double py3[size];

    for ( int i = 0; i <  size ; ++i ) {
       px[i] = i;
       py1[i] = size - i;
       py2[i] = size - 0.5 * i;
       py3[i] = size - 0.6 * i;
    }

    TGraph * gr1 = new TGraph( size, px, py1 );
    gr1->SetName("gr1");
    gr1->SetTitle("graph 1");
    gr1->SetMarkerStyle(21);
    gr1->SetDrawOption("AP");
    gr1->SetLineColor(2);
    gr1->SetLineWidth(4);
    gr1->SetFillStyle(0);

    TGraph * gr2 = new TGraph( size, px, py2 );
    gr2->SetName("gr2");
    gr2->SetTitle("graph 2");
    gr2->SetMarkerStyle(22);
    gr2->SetMarkerColor(2);
    gr2->SetDrawOption("P");
    gr2->SetLineColor(3);
    gr2->SetLineWidth(4);
    gr2->SetFillStyle(0);

    TGraph * gr3 = new TGraph( size, px, py3 );
    gr3->SetName("gr3");
    gr3->SetTitle("graph 3");
    gr3->SetMarkerStyle(23);
    gr3->SetLineColor(4);
    gr3->SetLineWidth(4);
    gr3->SetFillStyle(0);

    mg->Add( gr1 );
```

```
    mg->Add( gr2 );

    gr3->Draw("ALP");
    mg->Draw("LP");
    c3->BuildLegend();
```

TMultiLayerPerceptron

TNamed

继承 **TObject**

The TNamed class is the base class for all named ROOT classes.

A TNamed contains the essential elements (name, title) to identify a derived object in containers, directories and files. Most member functions defined in this base class are in general overridden by the derived classes.

class

```
   virtual void     Clear(Option_t *option="");//Set name and t
itle to empty strings ("")
   virtual TObject *Clone(const char *newname="") const;// Make
a clone of an object using the Streamer facility. If newname is
specified, this will be the name of the new object.
   virtual Int_t    Compare(const TObject *obj) const;// Compare
 two TNamed objects. Returns 0 when equal, -1 when this is small
er and +1 when bigger (like strcmp).
   virtual void     Copy(TObject &named) const;// Copy this to o
bj.
   virtual void     FillBuffer(char *&buffer);// Encode TNamed i
nto output buffer.
   virtual const char  *GetName() const { return fName; }
   virtual const char  *GetTitle() const { return fTitle; }
   virtual ULong_t  Hash() const { return fName.Hash(); }
   virtual Bool_t   IsSortable() const { return kTRUE; }
   virtual void     SetName(const char *name); // *MENU*   设置ob
ject identifier
/// Change (i.e. set) the name of the TNamed.
/// WARNING: if the object is a member of a THashTable or THashL
ist container
/// the container must be Rehash()'ed after SetName(). For examp
le the list
/// of objects in the current directory is a THashList.
```

```
   virtual void     SetNameTitle(const char *name, const char *t
itle);
/// Change (i.e. set) all the TNamed parameters (name and title).

/// WARNING: if the name is changed and the object is a member o
f a
/// THashTable or THashList container the container must be Reha
sh()'ed
/// after SetName(). For example the list of objects in the curr
ent
/// directory is a THashList.

   virtual void     SetTitle(const char *title=""); // *MENU*
设置object title
/// Change (i.e. set) the title of the TNamed.

   virtual void     ls(Option_t *option="") const;// List TNamed
 name and title.
   virtual void     Print(Option_t *option="") const;// Print TN
amed name and title.
   virtual Int_t    Sizeof() const;// Return size of the TNamed
part of the TObject.
```

TNeuron

TNtuple

继承 TTree

A simple tree with branches of floats.

A simple TTree restricted to a list of float variables only.

Each variable goes to a separate branch.

class

```
   TNtuple();
   TNtuple(const char *name,const char *title, const char *varli
st, Int_t bufsize=32000);
/// Create an Ntuple.
/// The parameter varlist describes the list of the ntuple varia
bles
/// separated by a colon:
/// Example:  `x:y:z:energy`
/// For each variable in the list a separate branch is created.
/// NOTE:
///  - Use TTree to create branches with variables of different
data types.
///  - Use TTree when the number of branches is large (> 100).

   virtual ~TNtuple();

   virtual void      Browse(TBrowser *b);/// Browse content of t
he ntuple
   virtual TTree    *CloneTree(Long64_t nentries = -1, Option_t*
 option = "");
/// Create a clone of this tree and copy nentries.
/// By default copy all entries.
/// Note that only active branches are copied.
/// The compression level of the cloned tree is set to the desti
nation file's
```

```
/// compression level.
/// See TTree::CloneTree for more details.

   virtual Int_t     Fill(const Float_t *x);/// Fill a Ntuple wi
th an array of floats
           Int_t     Fill(Int_t x0) { return Fill((Float_t)x0);
}
           Int_t     Fill(Double_t x0) { return Fill((Float_t)x0
); }
   virtual Int_t     Fill(Float_t x0, Float_t x1=0, Float_t x2=0
, Float_t x3=0,
                        Float_t x4=0, Float_t x5=0, Float_t x6=
0, Float_t x7=0,
                        Float_t x8=0, Float_t x9=0, Float_t x1
0=0,
                        Float_t x11=0, Float_t x12=0, Float_t
x13=0,
                        Float_t x14=0);/// Fill a Ntuple: Each
 Ntuple item is an argument
   virtual Int_t     GetNvar() const { return fNvar; }
           Float_t  *GetArgs() const { return fArgs; }
   virtual Long64_t  ReadStream(std::istream& inputStream, const
char *branchDescriptor="", char delimiter = ' ');
/// Read from filename as many columns as variables in the ntuple

/// the function returns the number of rows found in the file
/// The second argument "branchDescriptor" is currently not used.

/// Lines in the input file starting with "#" are ignored.

   virtual void      ResetBranchAddress(TBranch *);
/// Reset the branch addresses to the internal fArgs array. Use
this
/// method when the addresses were changed via calls to SetBranc
hAddress().

           void      ResetBranchAddresses();
/// Reset the branch addresses to the internal fArgs array. Use
this
/// method when the addresses were changed via calls to SetBranc
```

```
hAddress().
```

code

```
// A Ntuple is created via

TNtuple(name,title,varlist,bufsize)

// It is filled via:

TNtuple::Fill(*x)  or
TNtuple::Fill(v1,v2,v3.....)
```

example

TNtupleD

继承 TTree

A simple tree with branches of doubles.

class

```
   TNtupleD();
   TNtupleD(const char *name,const char *title, const char *varl
ist, Int_t bufsize=32000);
/// Create an Ntuple.
/// The parameter varlist describes the list of the ntuple varia
bles
/// separated by a colon:
/// Example:  `x:y:z:energy`
/// For each variable in the list a separate branch is created.
/// NOTE:
///  - Use TTree to create branches with variables of different
data types.
///  - Use TTree when the number of branches is large (> 100).

   virtual ~TNtupleD();

   virtual void      Browse(TBrowser *b);/// Browse content.
   virtual Int_t     Fill(const Double_t *x);/// Fill a Ntuple w
ith an array of floats.
   virtual Int_t     Fill(Double_t x0, Double_t x1, Double_t x2=0
, Double_t x3=0,
                           Double_t x4=0, Double_t x5=0, Double_t
 x6=0, Double_t x7=0,
                           Double_t x8=0, Double_t x9=0, Double_t
 x10=0,
                           Double_t x11=0, Double_t x12=0, Double
_t x13=0,
                           Double_t x14=0);/// Fill a Ntuple: Eac
h Ntuple item is an argument.
```

```
   virtual Int_t     GetNvar() const { return fNvar; }
           Double_t *GetArgs() const { return fArgs; }
   virtual Long64_t  ReadStream(std::istream& inputstream, const
char *branchDescriptor="", char delimiter = ' ');
/// Read from filename as many columns as variables in the ntuple

/// the function returns the number of rows found in the file
/// The second argument "branchDescriptor" is currently not used.

/// Lines in the input file starting with "#" are ignored.

   virtual void      ResetBranchAddress(TBranch *);
/// Reset the branch addresses to the internal fArgs array. Use
this
/// method when the addresses were changed via calls to SetBranc
hAddress().

           void      ResetBranchAddresses();
/// Reset the branch addresses to the internal fArgs array. Use
this
/// method when the addresses were changed via calls to SetBranc
hAddress().
```

code

example

TPad

继承 TVirtualPad, TAttBBox2D

The most important graphics class in the ROOT system.

A Pad is contained in a Canvas.

A Pad may contain other pads (unlimited pad hierarchy).

A pad is a linked list of primitives of any type (graphics objects, histograms, detectors, tracks, etc.).

Adding a new element into a pad is in general performed by the Draw member function of the object classes.

It is important to realize that the pad is a linked list of references to the original object. For example, in case of a histogram, the histogram.Draw() operation only stores a reference to the histogram object and not a graphical representation of this histogram. When the mouse is used to change (say the bin content), the bin content of the original histogram is changed.

The convention used in ROOT is that a Draw operation only adds a reference to the object. The effective drawing is performed when the canvas receives a signal to be painted.

In ExecuteEvent, move, changes can be performed on the object.

For examples of DistancetoPrimitive and ExecuteEvent functions, see classes

```
TLine::DistancetoPrimitive, TLine::ExecuteEvent
TBox::DistancetoPrimitive,  TBox::ExecuteEvent
TH1::DistancetoPrimitive,   TH1::ExecuteEvent
```

A Pad supports linear and log scales coordinate systems. The transformation coefficients are explained in TPad::ResizePad.

class

```
   // TPad status bits
   enum {
      kFraming      = BIT(6),
      kHori         = BIT(9),
      kClipFrame    = BIT(10),
      kPrintingPS   = BIT(11),
      kCannotMove   = BIT(12),
      kClearAfterCR = BIT(14)
   };

   TPad();
   TPad(const char *name, const char *title, Double_t xlow,
        Double_t ylow, Double_t xup, Double_t yup,
        Color_t color=-1, Short_t bordersize=-1, Short_t borderm
ode=-2);
/// Pad constructor.
///  A pad is a linked list of primitives.
///  A pad is contained in a canvas. It may contain other pads.
///  A pad has attributes. When a pad is created, the attributes
///  defined in the current style are copied to the pad attribut
es.
/// \param[in] name        pad name
/// \param[in] title       pad title
/// \param[in] xlow [0,1]  is the position of the bottom left po
int of the pad
///             expressed  in the mother pad reference system
/// \param[in] ylow [0,1]  is the Y position of this point.
/// \param[in] xup  [0,1]  is the x position of the top right po
int of the pad
///                        expressed in the mother pad reference
 system
/// \param[in] yup  [0,1]  is the Y position of this point.
/// \param[in] color       pad color
/// \param[in] bordersize  border size in pixels
/// \param[in] bordermode  border mode
///                        - bordermode = -1 box looks as it is
behind the screen
///                        - bordermode = 0  no special effects
///                        - bordermode = 1  box looks as it is
```

```
in front of the screen

   virtual ~TPad();
   void              AbsCoordinates(Bool_t set) { fAbsCoord = set
; }
   Double_t          AbsPixeltoX(Int_t px) {return fAbsPixeltoXk
 + px*fPixeltoX;}
   Double_t          AbsPixeltoY(Int_t py) {return fAbsPixeltoYk
 + py*fPixeltoY;}
   virtual void      AbsPixeltoXY(Int_t xpixel, Int_t ypixel, Do
uble_t &x, Double_t &y);
   virtual void      AddExec(const char *name, const char *comma
nd);
/// Add a new TExec object to the list of Execs.
/// When an event occurs in the pad (mouse click, etc) the list
of C++ commands
/// in the list of Execs are executed via TPad::AutoExec.
/// When a pad event occurs (mouse move, click, etc) all the com
mands
/// contained in the fExecs list are executed in the order found
 in the list.
/// This facility is activated by default. It can be deactivated
 by using
/// the canvas "Option" menu.
/// When moving the mouse in the canvas, a second canvas shows t
he
/// projection along X of the bin corresponding to the Y position

/// of the mouse. The resulting histogram is fitted with a gauss
ian.
/// A "dynamic" line shows the current bin position in Y.
/// This more elaborated example can be used as a starting point
/// to develop more powerful interactive applications exploiting
 the C++
/// interpreter as a development engine.

   virtual void      AutoExec();/// Execute the list of Execs wh
en a pad event occurs.
   virtual void      Browse(TBrowser *b);/// Browse pad.
   virtual TLegend  *BuildLegend(Double_t x1=0.5, Double_t y1=0.
```

```
67, Double_t x2=0.88, Double_t y2=0.88, const char *title=""); /
/ *MENU*
/// Build a legend from the graphical objects in the pad
/// A simple method to build automatically a TLegend from the
/// primitives in a TPad. Only those deriving from TAttLine,
/// TAttMarker and TAttFill are added, excluding TPave and TFrame

/// derived classes. x1, y1, x2, y2 are the TLegend coordinates.
/// title is the legend title. By default it is " ". The caller
/// program owns the returned TLegend.
/// If the pad contains some TMultiGraph or THStack the individu
al
/// graphs or histograms in them are added to the TLegend.

   TVirtualPad*      cd(Int_t subpadnumber=0); // *MENU*
/// Set Current pad.
/// When a canvas/pad is divided via TPad::Divide, one can direc
tly
/// set the current path to one of the subdivisions.
/// See TPad::Divide for the convention to number sub-pads.
/// Returns the new current pad, or 0 in case of failure.
///  Note1:  c1.cd() is equivalent to c1.cd(0) and sets the curr
ent pad
///          to c1 itself.
///  Note2:  after a statement like c1.cd(6), the global variabl
e gPad
///          points to the current pad. One can use gPad to set
attributes
///          of the current pad.
///  Note3:  One can get a pointer to one of the sub-pads of pad
 with:
///          TPad *subpad = (TPad*)pad->GetPad(subpadnumber);

   void              Clear(Option_t *option="");
/// Delete all pad primitives.
/// If the bit kClearAfterCR has been set for this pad, the Clea
r function
/// will execute only after having pressed a CarriageReturn
/// Set the bit with mypad->SetBit(TPad::kClearAfterCR)
```

```
   virtual Int_t     Clip(Float_t *x, Float_t *y, Float_t xclipl
, Float_t yclipb, Float_t xclipr, Float_t yclipt);
/// Clipping routine: Cohen Sutherland algorithm.
///  - If Clip ==2 the segment is outside the boundary.
///  - If Clip ==1 the segment has one point outside the boundar
y.
///  - If Clip ==0 the segment is inside the boundary.
/// \param[in]  x[],y[]                       Segment coordinate
s (2 points)
/// \param[in]  xclipl,yclipb,xclipr,yclipt   Clipping boundary
/// \param[out] x[],y[]                       New segment coordi
nates( 2 points)

   virtual Int_t     Clip(Double_t *x, Double_t *y, Double_t xcl
ipl, Double_t yclipb, Double_t xclipr, Double_t yclipt);
/// Clipping routine: Cohen Sutherland algorithm.
///  - If Clip ==2 the segment is outside the boundary.
///  - If Clip ==1 the segment has one point outside the boundar
y.
///  - If Clip ==0 the segment is inside the boundary.
/// \param[in]  x[],y[]                       Segment coordinate
s (2 points)
/// \param[in]  xclipl,yclipb,xclipr,yclipt   Clipping boundary
/// \param[out] x[],y[]                       New segment coordi
nates(2 points)

   virtual Int_t     ClippingCode(Double_t x, Double_t y, Double
_t xcl1, Double_t ycl1, Double_t xcl2, Double_t ycl2);/// Comput
e the endpoint codes for TPad::Clip.
   virtual Int_t     ClipPolygon(Int_t n, Double_t *x, Double_t
*y, Int_t nn, Double_t *xc, Double_t *yc, Double_t xclipl, Doubl
e_t yclipb, Double_t xclipr, Double_t yclipt);
/// Clip polygon using the Sutherland-Hodgman algorithm.
/// \param[in]  n                             Number of points in
 the polygon to
///                                          be clipped
/// \param[in]  x[n],y[n]                     Polygon do be clipp
ed vertices
/// \param[in]  xclipl,yclipb,xclipr,yclipt  Clipping boundary
/// \param[out] nn                            Number of points in
```

```
 xc and yc
/// \param[out] xc,yc                        Clipped polygon ver
tices. The Int_t
///                                          returned by this fu
nction is
///                                          the number of point
s in the clipped
///                                          polygon. These vect
ors must
///                                          be allocated by the
 calling function.
///                                          A size of 2*n for e
ach is
///                                          enough.
/// Sutherland and Hodgman's polygon-clipping algorithm uses a d
ivide-and-conquer
/// strategy: It solves a series of simple and identical problem
s that, when
/// combined, solve the overall problem. The simple problem is t
o clip a polygon
/// against a single infinite clip edge. Four clip edges, each d
efining one boundary
/// of the clip rectangle, successively clip a polygon against a
 clip rectangle.
///
/// Steps of Sutherland-Hodgman's polygon-clipping algorithm:
/// * Polygons can be clipped against each edge of the window on
e at a time.
///   Windows/edge intersections, if any, are easy to find since
 the X or Y coordinates
///   are already known.
/// * Vertices which are kept after clipping against one window
edge are saved for
///   clipping against the remaining edges.
/// * Note that the number of vertices usually changes and will
often increases.
/// The clip boundary determines a visible and invisible region.
 The edges from
/// vertex i to vertex i+1 can be one of four types:
/// * Case 1 : Wholly inside visible region - save endpoint
```

```
/// * Case 2 : Exit visible region - save the intersection
/// * Case 3 : Wholly outside visible region - save nothing
/// * Case 4 : Enter visible region - save intersection and endp
oint

   virtual void      Close(Option_t *option="");
/// Delete all primitives in pad and pad itself.
/// Pad cannot be used anymore after this call.
/// Emits signal Closed().

   virtual void      Closed() { Emit("Closed()"); } // *SIGNAL*
   virtual void      CopyPixmap();/// Copy the pixmap of the pad
 to the canvas.
   virtual void      CopyPixmaps();/// Copy the sub-pixmaps of t
he pad to the canvas.
   virtual void      DeleteExec(const char *name);/// Remove TEx
ec name from the list of Execs.
   virtual void      Divide(Int_t nx=1, Int_t ny=1, Float_t xmar
gin=0.01, Float_t ymargin=0.01, Int_t color=0); // *MENU*
/// Automatic pad generation by division.
///  - The current canvas is divided in nx by ny equal divisions
 (pads).
///  - xmargin is the space along x between pads in percent of c
anvas.
///  - ymargin is the space along y between pads in percent of c
anvas.
///  - color is the color of the new pads. If 0, color is the ca
nvas color.
/// Pads are automatically named canvasname_n where n is the div
ision number
/// starting from top left pad.

   virtual void      DivideSquare(Int_t n, Float_t xmargin=0.01,
 Float_t ymargin=0.01, Int_t color=0);
/// "n" is the total number of sub-pads. The number of sub-pads
along the X
/// and Y axis are computed according to the square root of n.

   virtual void      Draw(Option_t *option="");/// Draw Pad in C
urrent pad (re-parent pad if necessary).
```

```
   virtual void      DrawClassObject(const TObject *obj, Option_
t *option="");
/// Draw class inheritance tree of the class to which obj belong
s.
/// If a class B inherits from a class A, description of B is dr
awn
/// on the right side of description of A.
/// Member functions overridden by B are shown in class A with a
 blue line
/// crossing-out the corresponding member function.

   static  void      DrawColorTable();/// Static function to Dis
play Color Table in a pad.

  virtual void      DrawCrosshair();
/// Function called to draw a crosshair in the canvas
/// When moving the mouse in the canvas, a crosshair is drawn
///  - if the canvas fCrosshair = 1 , the crosshair spans the fu
ll canvas
///  - if the canvas fCrosshair > 1 , the crosshair spans only t
he pad

   TH1F             *DrawFrame(Double_t xmin, Double_t ymin, Dou
ble_t xmax, Double_t ymax, const char *title="");
///  Draw an empty pad frame with X and Y axis.
///   \param[in] xmin      X axis lower limit
///   \param[in] xmax      X axis upper limit
///   \param[in] ymin      Y axis lower limit
///   \param[in] ymax      Y axis upper limit
///   \param[in] title     Pad title.If title is of the form "st
ringt;stringx;stringy"
///                        the pad title is set to stringt, the
x axis title to
///                        stringx, the y axis title to stringy.

   virtual void      ExecuteEventAxis(Int_t event, Int_t px, Int
_t py, TAxis *axis);
/// Execute action corresponding to one event for a TAxis object
 (called by TAxis::ExecuteEvent.)
///  This member function is called when an axis is clicked with
```

```
 the locator
/// The axis range is set between the position where the mouse i
s pressed
/// and the position where it is released.
/// If the mouse position is outside the current axis range when
 it is released
/// the axis is unzoomed with the corresponding proportions.
/// Note that the mouse does not need to be in the pad or even c
anvas
/// when it is released.

   virtual TObject  *FindObject(const char *name) const;
/// Search if object named name is inside this pad or in pads in
side this pad.
/// In case name is in several sub-pads the first one is returne
d.

   virtual TObject  *FindObject(const TObject *obj) const;
/// Search if obj is in pad or in pads inside this pad.
/// In case obj is in several sub-pads the first one is returned.


   virtual void      UseCurrentStyle();  // *MENU*
/// Force a copy of current style for all objects in pad.

   virtual Short_t   GetBorderMode() const { return fBorderMode;
}
   virtual Short_t   GetBorderSize() const { return fBorderSize;
}
   Int_t             GetCrosshair() const;
/// Return the crosshair type (from the mother canvas)
/// crosshair type = 0 means no crosshair.

   virtual Int_t     GetCanvasID() const;/// Get canvas identifi
er.
   virtual TCanvasImp *GetCanvasImp() const;/// Get canvas imple
mentation pointer if any
   TFrame           *GetFrame();/// Get frame.
   virtual Int_t     GetEvent() const;/// Get Event.
   virtual Int_t     GetEventX() const;/// Get X event.
```

```
   virtual Int_t      GetEventY() const;/// Get Y event.
   virtual Color_t    GetHighLightColor() const;/// Get highlight
 color.
   virtual void       GetRange(Double_t &x1, Double_t &y1, Double
_t &x2, Double_t &y2);/// Return pad world coordinates range.
   virtual void       GetRangeAxis(Double_t &xmin, Double_t &ymin
, Double_t &xmax, Double_t &ymax);/// Return pad axis coordinate
s range.
   virtual void       GetPadPar(Double_t &xlow, Double_t &ylow, D
ouble_t &xup, Double_t &yup);/// Return lower and upper bounds o
f the pad in NDC coordinates.
   Double_t           GetXlowNDC() const {return fXlowNDC;}
   Double_t           GetYlowNDC() const {return fYlowNDC;}
   Double_t           GetWNDC() const {return fWNDC;}
   Double_t           GetHNDC() const {return fHNDC;}
   virtual UInt_t     GetWw() const;/// Get Ww.
   virtual UInt_t     GetWh() const;/// Get Wh.
   Double_t           GetAbsXlowNDC() const {return fAbsXlowNDC;}
   Double_t           GetAbsYlowNDC() const {return fAbsYlowNDC;}
   Double_t           GetAbsWNDC() const {return fAbsWNDC;}
   Double_t           GetAbsHNDC() const {return fAbsHNDC;}
   Double_t           GetAspectRatio() const { return fAspectRati
o; }
   Double_t           GetPhi() const   {return fPhi;}
   Double_t           GetTheta() const {return fTheta;}
   Double_t           GetUxmin() const {return fUxmin;}
   Double_t           GetUymin() const {return fUymin;}
   Double_t           GetUxmax() const {return fUxmax;}
   Double_t           GetUymax() const {return fUymax;}
   Bool_t             GetGridx() const {return fGridx;}
   Bool_t             GetGridy() const {return fGridy;}
   Int_t              GetNumber() const {return fNumber;}
   Int_t              GetTickx() const {return fTickx;}
   Int_t              GetTicky() const {return fTicky;}
   Double_t           GetX1() const { return fX1; }
   Double_t           GetX2() const { return fX2; }
   Double_t           GetY1() const { return fY1; }
   Double_t           GetY2() const { return fY2; }
   static Int_t       GetMaxPickDistance();/// Static function (s
ee also TPad::SetMaxPickDistance)
```

```
   TList            *GetListOfPrimitives() const {return fPrimit
ives;}
   TList            *GetListOfExecs() const {return fExecs;}
   virtual TObject  *GetPrimitive(const char *name) const;  //ob
solete, use FindObject instead
   virtual TObject  *GetSelected() const;/// Get selected.
   virtual TVirtualPad  *GetPad(Int_t subpadnumber) const;/// Ge
t a pointer to subpadnumber of this pad.
   virtual TObject  *GetPadPointer() const {return fPadPointer;}
   TVirtualPad      *GetPadSave() const;/// Get save pad.
   TVirtualPad      *GetSelectedPad() const;/// Get selected pad.

   Int_t             GetGLDevice();/// Get GL device.
   TView            *GetView() const {return fView;}
   TObject          *GetView3D() const {return fPadView3D;}// Re
turn 3D View of this TPad
   Int_t             GetLogx() const {return fLogx;}
   Int_t             GetLogy() const {return fLogy;}
   Int_t             GetLogz() const {return fLogz;}
   virtual TVirtualPad *GetMother() const {return fMother;}
   const char       *GetName() const {return fName.Data();}
   const char       *GetTitle() const {return fTitle.Data();}
   virtual TCanvas  *GetCanvas() const { return fCanvas; }
   virtual TVirtualPad *GetVirtCanvas() const ;/// Get virtual c
anvas.
   virtual TVirtualPadPainter *GetPainter();/// Get pad painter
from TCanvas.
   Int_t             GetPadPaint() const {return fPadPaint;}
   Int_t             GetPixmapID() const {return fPixmapID;}
   ULong_t           Hash() const { return fName.Hash(); }
   virtual Bool_t    HasCrosshair() const;/// Return kTRUE if th
e crosshair has been activated (via SetCrosshair).
   void              HighLight(Color_t col=kRed, Bool_t set=kTRU
E);
/// Highlight pad.
/// do not highlight when printing on Postscript

   Bool_t            HasFixedAspectRatio() const { return fFixed
AspectRatio; }
   virtual Bool_t    IsBatch() const;/// Is pad in batch mode ?
```

```
   virtual Bool_t    IsEditable() const {return fEditable;}
   Bool_t            IsFolder() const {return kTRUE;}
   Bool_t            IsModified() const {return fModified;}
   virtual Bool_t    IsRetained() const;/// Is pad retained ?
   virtual Bool_t    IsVertical() const {return !TestBit(kHori);
}
   virtual void      ls(Option_t *option="") const;/// List all
primitives in pad.
   void              Modified(Bool_t flag=1);  // *SIGNAL*   //
 Set to true when pad is modified
   virtual Bool_t    OpaqueMoving() const;/// Is pad moving in o
paque mode ?
   virtual Bool_t    OpaqueResizing() const;/// Is pad resizing
in opaque mode ?
   Double_t          PadtoX(Double_t x) const;/// Convert x from
 pad to X.
   Double_t          PadtoY(Double_t y) const;/// Convert y from
 pad to Y.
   virtual void      Paint(Option_t *option="");/// Paint all pr
imitives in pad.
   void              PaintBox(Double_t x1, Double_t y1, Double_t
 x2, Double_t y2, Option_t *option="");
/// Paint box in CurrentPad World coordinates.
///  - if option[0] = 's' the box is forced to be paint with sty
le=0
///  - if option[0] = 'l' the box contour is drawn

   void              PaintFillArea(Int_t n, Float_t *x, Float_t
*y, Option_t *option=""); // Obsolete
   void              PaintFillArea(Int_t n, Double_t *x, Double_
t *y, Option_t *option=""); /// Paint fill area in CurrentPad Wo
rld coordinates.
   void              PaintFillAreaHatches(Int_t n, Double_t *x,
Double_t *y, Int_t FillStyle);
/// This function paints hatched fill area according to the Fill
Style value
/// The convention for the Hatch is the following:
///     FillStyle = 3ijk
///  -  i (1-9) : specify the space between each hatch
///             1 = minimum  9 = maximum
```

```
///              the final spacing is i*GetHatchesSpacing(). The
hatches spacing
///              is set by SetHatchesSpacing()
///  -  j (0-9) : specify angle between 0 and 90 degrees
///              * 0 = 0
///              * 1 = 10
///              * 2 = 20
///              * 3 = 30
///              * 4 = 45
///              * 5 = Not drawn
///              * 6 = 60
///              * 7 = 70
///              * 8 = 80
///              * 9 = 90
///  -  k (0-9) : specify angle between 90 and 180 degrees
///              * 0 = 180
///              * 1 = 170
///              * 2 = 160
///              * 3 = 150
///              * 4 = 135
///              * 5 = Not drawn
///              * 6 = 120
///              * 7 = 110
///              * 8 = 100
///              * 9 = 90

   void              PaintHatches(Double_t dy, Double_t angle, I
nt_t nn, Double_t *xx, Double_t *yy);
/// This routine draw hatches inclined with the
/// angle "angle" and spaced of "dy" in normalized device
/// coordinates in the surface defined by n,xx,yy.

   void              PaintPadFrame(Double_t xmin, Double_t ymin,
 Double_t xmax, Double_t ymax);/// Paint histogram/graph frame.
   void              PaintLine(Double_t x1, Double_t y1, Double_
t x2, Double_t y2);/// Paint line in CurrentPad World coordinate
s.
   void              PaintLineNDC(Double_t u1, Double_t v1,Doubl
e_t u2, Double_t v2);/// Paint line in normalized coordinates.
   void              PaintLine3D(Float_t *p1, Float_t *p2);/// P
```

```
aint 3-D line in the CurrentPad.
   void              PaintLine3D(Double_t *p1, Double_t *p2);///
 Paint 3-D line in the CurrentPad.
   void              PaintPolyLine(Int_t n, Float_t *x, Float_t
*y, Option_t *option="");
/// Paint polyline in CurrentPad World coordinates.

   void              PaintPolyLine(Int_t n, Double_t *x, Double_
t *y, Option_t *option="");
/// Paint polyline in CurrentPad World coordinates.
///  If option[0] == 'C' no clipping

   void              PaintPolyLine3D(Int_t n, Double_t *p);/// P
aint 3-D polyline in the CurrentPad.
   void              PaintPolyLineNDC(Int_t n, Double_t *x, Doub
le_t *y, Option_t *option="");
/// Paint polyline in CurrentPad NDC coordinates.

   void              PaintPolyMarker(Int_t n, Float_t *x, Float_
t *y, Option_t *option="");
/// Paint polymarker in CurrentPad World coordinates.

   void              PaintPolyMarker(Int_t n, Double_t *x, Doubl
e_t *y, Option_t *option="");
/// Paint polymarker in CurrentPad World coordinates.

   virtual void      PaintModified();
   void              PaintText(Double_t x, Double_t y, const char
 *text);/// Paint text in CurrentPad World coordinates.
   void              PaintText(Double_t x, Double_t y, const wch
ar_t *text);/// Paint text in CurrentPad World coordinates.
   void              PaintTextNDC(Double_t u, Double_t v, const
char *text);/// Paint text in CurrentPad NDC coordinates.
   void              PaintTextNDC(Double_t u, Double_t v, const
wchar_t *text);/// Paint text in CurrentPad NDC coordinates.
   virtual TPad     *Pick(Int_t px, Int_t py, TObjLink *&pickobj)
;
/// Search for an object at pixel position px,py.
///  Check if point is in this pad.
///  If yes, check if it is in one of the sub-pads
```

```
///  If found in the pad, compute closest distance of approach t
o each primitive.
///  If one distance of approach is found to be within the limit
 Distancemaximum
///  the corresponding primitive is selected and the routine ret
urns.

   Double_t          PixeltoX(Int_t px);
   Double_t          PixeltoY(Int_t py);
   virtual void      PixeltoXY(Int_t xpixel, Int_t ypixel, Doubl
e_t &x, Double_t &y);
   virtual void      Pop();  // *MENU* /// Pop pad to the top of
 the stack.
   virtual void      Print(const char *filename="") const;
/// Save Pad contents in a file in one of various formats.
///  - if filename is "", the file produced is padname.ps
///  - if filename starts with a dot, the padname is added in fr
ont
///  - if filename contains .eps, an Encapsulated Postscript fil
e is produced
///  - if filename contains .gif, a GIF file is produced
///  - if filename contains .gif+NN, an animated GIF file is pro
duced
///       See comments in TASImage::WriteImage for meaning of NN
 and other
///       .gif suffix variants
///  - if filename contains .C or .cxx, a C++ macro file is prod
uced
///  - if filename contains .root, a Root file is produced
///  - if filename contains .xml,  a XML file is produced
///  See comments in TPad::SaveAs or the TPad::Print function be
low

   virtual void      Print(const char *filename, Option_t *optio
n);
/// Save Canvas contents in a file in one of various formats.
/// option can be:
///       -          0   - as "ps"
///       -        "ps"  - Postscript file is produced (see spec
ial cases below)
```

```
///       -   "Portrait" - Postscript file is produced (Portrait)

///       -  "Landscape" - Postscript file is produced (Landscap
e)
///       -     "Title:" - The character string after "Title:" b
ecomes a table
///       -                of content entry (for PDF files).
///       -        "eps" - an Encapsulated Postscript file is pr
oduced
///       -    "Preview" - an Encapsulated Postscript file with
preview is produced.
///       -        "pdf" - a PDF file is produced
///       -        "svg" - a SVG file is produced
///       -        "tex" - a TeX file is produced
///       -        "gif" - a GIF file is produced
///       -     "gif+NN" - an animated GIF file is produced, whe
re NN is delay in 10ms units NOTE: See other variants for loopin
g animation in TASImage::WriteImage
///       -        "xpm" - a XPM file is produced
///       -        "png" - a PNG file is produced
///       -        "jpg" - a JPEG file is produced. NOTE: JPEG's
 lossy compression will make all sharp edges fuzzy.
///        -      "tiff" - a TIFF file is produced
///        -       "cxx" - a C++ macro file is produced
///        -       "xml" - a XML file
///        -      "root" - a ROOT binary file
///     filename = 0 - filename  is defined by the GetName and i
ts
///                     extension is defined with the option
/// When Postscript output is selected (ps, eps), the canvas is
saved
/// to filename.ps or filename.eps. The aspect ratio of the canv
as is preserved
/// on the Postscript file. When the "ps" option is selected, th
e Postscript
/// page will be landscape format if the canvas is in landscape
format, otherwise
/// portrait format is selected.
/// The physical size of the Postscript page is the one selected
 in the
```

```
/// current style. This size can be modified via TStyle::SetPape
rSize.

   virtual void      Range(Double_t x1, Double_t y1, Double_t x2
, Double_t y2); // *MENU* *ARGS={x1=>fX1,y1=>fY1,x2=>fX2,y2=>fY2}

/// Set world coordinate system for the pad.
/// Emits signal "RangeChanged()", in the slot get the range
/// via GetRange().

   virtual void      RangeChanged() { Emit("RangeChanged()"); }
// *SIGNAL*
   virtual void      RangeAxis(Double_t xmin, Double_t ymin, Dou
ble_t xmax, Double_t ymax);
/// Set axis coordinate system for the pad.
/// The axis coordinate system is a subset of the world coordina
te system
/// xmin,ymin is the origin of the current coordinate system,
/// xmax is the end of the X axis, ymax is the end of the Y axis.

/// By default a margin of 10 per cent is left on all sides of t
he pad
/// Emits signal "RangeAxisChanged()", in the slot get the axis
range
/// via GetRangeAxis().

   virtual void      RangeAxisChanged() { Emit("RangeAxisChanged
()"); } // *SIGNAL*
   virtual void      RecursiveRemove(TObject *obj);/// Recursive
ly remove object from a pad and its sub-pads.
   virtual void      RedrawAxis(Option_t *option="");
///  Redraw the frame axis
///  Redrawing axis may be necessary in case of superimposed his
tograms
///  when one or more histograms have a fill color
///  Instead of calling this function, it may be more convenient
///  to call directly h1->Draw("sameaxis") where h1 is the point
er
///  to the first histogram drawn in the pad.
///  By default, if the pad has the options gridx or/and gridy a
```

```
ctivated,
///  the grid is not drawn by this function.
///  if option="g" is specified, this will force the drawing of
the grid
///  on top of the picture

   virtual void      ResetView3D(TObject *view=0){fPadView3D=vie
w;}
   virtual void      ResizePad(Option_t *option="");/// Compute
pad conversion coefficients.
   virtual void      SaveAs(const char *filename="",Option_t *op
tion="") const; // *MENU*
/// Save Pad contents in a file in one of various formats.
///  - if filename is "", the file produced is padname.ps
///  - if filename starts with a dot, the padname is added in fr
ont
///  - if filename contains .eps, an Encapsulated Postscript fil
e is produced
///  - if filename contains .pdf, a PDF file is produced
///  - if filename contains .svg, a SVG file is produced
///  - if filename contains .tex, a TeX file is produced
///  - if filename contains .gif, a GIF file is produced
///  - if filename contains .gif+NN, an  animated GIF file is pr
oduced See comments in TASImage::WriteImage for meaning of NN an
d other .gif sufix variants
///  - if filename contains .xpm, a XPM file is produced
///  - if filename contains .png, a PNG file is produced
///  - if filename contains .jpg, a JPEG file is produced NOTE:
JPEG's lossy compression will make all sharp edges fuzzy.
///  - if filename contains .tiff, a TIFF file is produced
///  - if filename contains .C or .cxx, a C++ macro file is prod
uced
///  - if filename contains .root, a Root file is produced
///  - if filename contains .xml, a XML file is produced
///   See comments in TPad::Print for the Postscript formats

   virtual void      SetBorderMode(Short_t bordermode) {fBorderM
ode = bordermode; Modified();} // *MENU*
   virtual void      SetBorderSize(Short_t bordersize) {fBorderS
ize = bordersize; Modified();} // *MENU*
```

```
   void              SetCanvas(TCanvas *c) { fCanvas = c; }
   virtual void      SetCanvasSize(UInt_t ww, UInt_t wh);/// Set
 canvas size.
   virtual void      SetCrosshair(Int_t crhair=1); // *TOGGLE*
/// Set crosshair active/inactive.
///  - If crhair != 0, a crosshair will be drawn in the pad and
its sub-pads.
///  - If the canvas crhair = 1 , the crosshair spans the full c
anvas.
///  - If the canvas crhair > 1 , the crosshair spans only the p
ad.

   virtual void      SetCursor(ECursor cursor);/// Set cursor ty
pe.
   virtual void      SetDoubleBuffer(Int_t mode=1);/// Set doubl
e buffer mode ON or OFF.
   virtual void      SetDrawOption(Option_t *option="");
   virtual void      SetEditable(Bool_t mode=kTRUE); // *TOGGLE*
/// Set pad editable yes/no
/// If a pad is not editable:
/// - one cannot modify the pad and its objects via the mouse.
/// - one cannot add new objects to the pad

   virtual void      SetFixedAspectRatio(Bool_t fixed = kTRUE);
// *TOGGLE* /// Fix pad aspect ratio to current value if fixed i
s true.
   virtual void      SetGrid(Int_t valuex = 1, Int_t valuey = 1)
{fGridx = valuex; fGridy = valuey; Modified();}
   virtual void      SetGridx(Int_t value = 1) {fGridx = value;
Modified();} // *TOGGLE*
   virtual void      SetGridy(Int_t value = 1) {fGridy = value;
Modified();} // *TOGGLE*
   virtual void      SetFillStyle(Style_t fstyle);
/// Override TAttFill::FillStyle for TPad because we want to han
dle style=0
/// as style 4000.

   virtual void      SetLogx(Int_t value = 1); // *TOGGLE*
/// Set Lin/Log scale for X
///  - value = 0 X scale will be linear
```

```
///  - value = 1 X scale will be logarithmic (base 10)
///  - value > 1 reserved for possible support of base e or other


   virtual void      SetLogy(Int_t value = 1); // *TOGGLE*
/// Set Lin/Log scale for Y
///  - value = 0 Y scale will be linear
///  - value = 1 Y scale will be logarithmic (base 10)
///  - value > 1 reserved for possible support of base e or other


   virtual void      SetLogz(Int_t value = 1); // *TOGGLE* /// S
et Lin/Log scale for Z
   virtual void      SetNumber(Int_t number) {fNumber = number;}
   virtual void      SetPad(const char *name, const char *title,
                           Double_t xlow, Double_t ylow, Double_
t xup,
                           Double_t yup, Color_t color=35,
                           Short_t bordersize=5, Short_t borderm
ode=-1);/// Set all pad parameters.
   virtual void      SetPad(Double_t xlow, Double_t ylow, Double
_t xup, Double_t yup);
/// Set canvas range for pad and resize the pad. If the aspect r
atio
/// was fixed before the call it will be un-fixed.

   virtual void      SetAttFillPS(Color_t color, Style_t style);
/// Set postscript fill area attributes.
   virtual void      SetAttLinePS(Color_t color, Style_t style,
Width_t lwidth);/// Set postscript line attributes.
   virtual void      SetAttMarkerPS(Color_t color, Style_t style
, Size_t msize);/// Set postscript marker attributes.
   virtual void      SetAttTextPS(Int_t align, Float_t angle, Co
lor_t color, Style_t font, Float_t tsize);/// Set postscript tex
t attributes.
   static  void      SetMaxPickDistance(Int_t maxPick=5);
/// static function to set the maximum Pick Distance fgMaxPickDi
stance
/// This parameter is used in TPad::Pick to select an object if
/// its DistancetoPrimitive returns a value < fgMaxPickDistance
```

```
/// The default value is 5 pixels. Setting a smaller value will
make
/// picking more precise but also more difficult

   virtual void      SetName(const char *name) {fName = name;} /
/ *MENU*
   virtual void      SetSelected(TObject *obj);/// Set selected.
   virtual void      SetTicks(Int_t valuex = 1, Int_t valuey = 1)
 {fTickx = valuex; fTicky = valuey; Modified();}
   virtual void      SetTickx(Int_t value = 1) {fTickx = value;
Modified();} // *TOGGLE*
   virtual void      SetTicky(Int_t value = 1) {fTicky = value;
Modified();} // *TOGGLE*
   virtual void      SetTitle(const char *title="") {fTitle = ti
tle;}
   virtual void      SetTheta(Double_t theta=30) {fTheta = theta
; Modified();}
   virtual void      SetPhi(Double_t phi=30) {fPhi = phi; Modifi
ed();}
   virtual void      SetToolTipText(const char *text, Long_t del
ayms = 1000);
/// Set tool tip text associated with this pad. The delay is in
/// milliseconds (minimum 250). To remove tool tip call method w
ith
/// text = 0.

   virtual void      SetVertical(Bool_t vert=kTRUE);/// Set pad
vertical (default) or horizontal
   virtual void      SetView(TView *view = 0);/// Set the curren
t TView. Delete previous view if view=0
   virtual void      SetViewer3D(TVirtualViewer3D *viewer3d) {fV
iewer3D = viewer3d;}

   virtual void      SetGLDevice(Int_t dev) {fGLDevice = dev;}
   virtual void      SetCopyGLDevice(Bool_t copy) {fCopyGLDevice
 = copy;}

   virtual void      ShowGuidelines(TObject *object, const Int_t
 event, const char mode = 'i', const bool cling = true);
/// Shows lines to indicate if a TAttBBox2D object is aligned to
```

```
/// the center or to another object, shows distance arrows if two

/// objects on screen have the same distance to another object
/// Call from primitive in Execute Event, in ButtonMotion after
/// the new coordinates have been set, to 'stick'
/// once when button is up to delete lines
/// modes: t (Top), b (bottom), l (left), r (right), i (inside)
/// in resize modes (t,b,l,r) only size arrows are sticky
/// in mode, the function gets the point on the element that is
clicked to
/// move (i) or resize (all others).

   virtual void      Update();/// Update pad.
   Int_t             UtoAbsPixel(Double_t u) const {return Int_t
(fUtoAbsPixelk + u*fUtoPixel);}
   Int_t             VtoAbsPixel(Double_t v) const {return Int_t
(fVtoAbsPixelk + v*fVtoPixel);}
   Int_t             UtoPixel(Double_t u) const;
   Int_t             VtoPixel(Double_t v) const;
   virtual TObject  *WaitPrimitive(const char *pname="", const c
har *emode="");
/// Loop and sleep until a primitive with name=pname is found in
 the pad.
/// If emode is given, the editor is automatically set to emode,
 ie
/// it is not required to have the editor control bar.
/// The possible values for emode are:
///  - emode = "" (default). User will select the mode via the e
ditor bar
///  - emode = "Arc", "Line", "Arrow", "Button", "Diamond", "Ell
ipse",
///  - emode = "Pad","pave", "PaveLabel","PaveText", "PavesText",

///  - emode = "PolyLine", "CurlyLine", "CurlyArc", "Text", "Mar
ker", "CutG"
/// If emode is specified and it is not valid, "PolyLine" is ass
umed. If emode
/// is not specified or ="", an attempt is to use pname[1...]
///
/// for example if pname="TArc", emode="Arc" will be assumed.
```

```
/// When this function is called within a macro, the macro execu
tion
/// is suspended until a primitive corresponding to the arguments

/// is found in the pad.
/// If CRTL/C is typed in the pad, the function returns 0.
/// While this function is executing, one can use the mouse, int
eract
/// with the graphics pads, use the Inspector, Browser, TreeView
er, etc.
/// Examples:
///   c1.WaitPrimitive();      // Return the first created primi
tive
///                            // whatever it is.
///                            // If a double-click with the mou
se is executed
///                            // in the pad or any key pressed,
 the function
///                            // returns 0.
///   c1.WaitPrimitive("ggg"); // Set the editor in mode "PolyLi
ne/Graph"
///                            // Create a polyline, then using
the context
///                            // menu item "SetName", change th
e name
///                            // of the created TGraph to "ggg"
///   c1.WaitPrimitive("TArc");// Set the editor in mode "Arc".
Returns
///                            // as soon as a TArc object is cr
eated.
///   c1.WaitPrimitive("lat","Text"); // Set the editor in Text/
Latex mode.
///                            // Create a text object, then Set
 its name to "lat"

   Int_t             XtoAbsPixel(Double_t x) const;
   Int_t             YtoAbsPixel(Double_t y) const;
   Double_t          XtoPad(Double_t x) const;/// Convert x from
 X to pad.
   Double_t          YtoPad(Double_t y) const;/// Convert y from
```

```
 Y to pad.
   Int_t             XtoPixel(Double_t x) const;
   Int_t             YtoPixel(Double_t y) const;
   virtual void      XYtoAbsPixel(Double_t x, Double_t y, Int_t
&xpixel, Int_t &ypixel) const;
   virtual void      XYtoPixel(Double_t x, Double_t y, Int_t &xp
ixel, Int_t &ypixel) const;

   virtual TObject  *CreateToolTip(const TBox *b, const char *te
xt, Long_t delayms);/// Create a tool tip and return its pointer.

   virtual void      DeleteToolTip(TObject *tip);/// Delete tool
 tip object.
   virtual void      ResetToolTip(TObject *tip);
/// Reset tool tip, i.e. within time specified in CreateToolTip
the
/// tool tip will pop up.

   virtual void      CloseToolTip(TObject *tip);/// Hide tool ti
p.

   virtual void      x3d(Option_t *type=""); // Depreciated
/// Deprecated: use TPad::GetViewer3D() instead

   virtual TVirtualViewer3D *GetViewer3D(Option_t * type = "");
/// Create/obtain handle to 3D viewer. Valid types are:
///  - 'pad' - pad drawing via TViewer3DPad
/// any others registered with plugin manager supporting TVirtua
lViewer3D
/// If an invalid/null type is requested then the current viewer
 is returned
/// (if any), otherwise a default 'pad' type is returned

   virtual Bool_t            HasViewer3D() const { return (fView
er3D); }
   virtual void              ReleaseViewer3D(Option_t * type = ""
);/// Release current (external) viewer

   virtual Rectangle_t  GetBBox();/// Return the bounding Box of
 the Pad
```

```
   virtual TPoint       GetBBoxCenter();/// Return the center of
 the Pad as TPoint in pixels

   virtual void         SetBBoxCenter(const TPoint &p);/// Set c
enter of the Pad

   virtual void         SetBBoxCenterX(const Int_t x);
/// Set X coordinate of the center of the Pad

   virtual void         SetBBoxCenterY(const Int_t y);
/// Set Y coordinate of the center of the Pad

   virtual void         SetBBoxX1(const Int_t x);
/// Set lefthandside of BoundingBox to a value
/// (resize in x direction on left)

   virtual void         SetBBoxX2(const Int_t x);
/// Set right hand side of BoundingBox to a value
/// (resize in x direction on right)

   virtual void         SetBBoxY1(const Int_t y);
/// Set top of BoundingBox to a value (resize in y direction on
top)

   virtual void         SetBBoxY2(const Int_t y);
/// Set bottom of BoundingBox to a value
/// (resize in y direction on bottom)

   virtual void      RecordPave(const TObject *obj);
  // *SIGNAL*
/// Emit RecordPave() signal.

   virtual void      RecordLatex(const TObject *obj);
  // *SIGNAL*
/// Emit RecordLatex() signal.

   virtual void      EventPave() { Emit("EventPave()"); }
  // *SIGNAL*
   virtual void      StartEditing() { Emit("StartEditing()"); }
  // *SIGNAL*
```

code

```
//图片修饰
gPad->SetTickx(1);//上边框有刻度
gPad->SetTicky(1);//右边框有刻度
gPad->SetTickx(2);//上边框有刻度和数值
gPad->SetTicky(2);//右边框有刻度和数值



TPad* pad1 = new TPad("pad1","pad1",0.03,0.62,0.50,0.92,32);//x起点，y起点，x终点，y终点，颜色
pad1->Draw();
pad1->cd();
pad1->SetLogy();//y轴 对数坐标
pad1->SetGridy();//y轴 网格
pad2->SetLogx();
pad2->SetGridx();
```

```
/// The following macro waits for 10 primitives of any type to be created.

   TCanvas c1("c1");
   TObject *obj;
   for (Int_t i=0;i<10;i++) {
      obj = gPad->WaitPrimitive();
      if (!obj) break;
      printf("Loop i=%d, found objIsA=%s, name=%s\n",
         i,obj->ClassName(),obj->GetName());
   }
```

```
/// Function called to draw a crosshair in the canvas
/// When moving the mouse in the canvas, a crosshair is drawn
///  - if the canvas fCrosshair = 1 , the crosshair spans the fu
ll canvas
///  - if the canvas fCrosshair > 1 , the crosshair spans only t
he pad

TFile f("hsimple.root");
hpxpy.Draw();
c1.SetCrosshair();
```

```
///  The following examples of TExec commands are provided in th
e tutorials:

Root > TFile f("hsimple.root")
Root > hpx.Draw()
Root > c1.AddExec("ex1",".x exec1.C")

/// At this point you can use the mouse to click on the contour
of
/// the histogram hpx. When the mouse is clicked, the bin number
 and its
/// contents are printed.

Root > TFile f("hsimple.root")
Root > hpxpy.Draw()
Root > c1.AddExec("ex2",".x exec2.C")
```

```
/// The physical size of the Postscript page is the one selected
 in the
/// current style. This size can be modified via TStyle::SetPape
rSize.

gStyle->SetPaperSize(TStyle::kA4);  //default
gStyle->SetPaperSize(TStyle::kUSLetter);

/// where TStyle::kA4 and TStyle::kUSLetter are defined in the e
```

```
num
/// EPaperSize in TStyle.h

/// An alternative is to call:

gStyle->SetPaperSize(20,26);  same as kA4
gStyle->SetPaperSize(20,24);  same as kUSLetter

///   The above numbers take into account some margins and are i
n centimeters.

///  The "Preview" option allows to generate a preview (in the T
IFF format) within
///  the Encapsulated Postscript file. This preview can be used
by programs like
///  MSWord to visualize the picture on screen. The "Preview" op
tion relies on the
///  epstool command (http://www.cs.wisc.edu/~ghost/gsview/epsto
ol.htm).

canvas->Print("example.eps","Preview");

///  To generate a Postscript file containing more than one pict
ure, see
///  class TPostScript.
///
/// ### Writing several canvases to the same Postscript or PDF f
ile:
///
///  - if the Postscript or PDF file name finishes with "(", the
 file is not closed
///  - if the Postscript or PDF file name finishes with ")" and
the file has been opened
///    with "(", the file is closed.

TCanvas c1("c1");
h1.Draw();
c1.Print("c1.ps("); //write canvas and keep the ps file open
h2.Draw();
c1.Print("c1.ps"); canvas is added to "c1.ps"
```

```
h3.Draw();
c1.Print("c1.ps)"); canvas is added to "c1.ps" and ps file is cl
osed


/// In the previous example replacing "ps" by "pdf" will create
a multi-pages PDF file.

/// Note that the following sequence writes the canvas to "c1.ps
" and closes the ps file.:

TCanvas c1("c1");
h1.Draw();
c1.Print("c1.ps");

///  The TCanvas::Print("file.ps(") mechanism is very useful, bu
t it can be
///  a little inconvenient to have the action of opening/closing
 a file
///  being atomic with printing a page. Particularly if pages ar
e being
///  generated in some loop one needs to detect the special case
s of first
///  and last page and then munge the argument to Print() accord
ingly.
///
///  The "[" and "]" can be used instead of "(" and ")".

c1.Print("file.ps[");   // No actual print, just open file.ps
for (int i=0; i<10; ++i) {
  // fill canvas for context i
  // ...

  c1.Print("file.ps");  // actually print canvas to file
}// end loop
c1.Print("file.ps]");   // No actual print, just close.


/// As before, the same macro is valid for PDF files.
///
```

```
/// It is possible to print a canvas into an animated GIF file b
y specifying the
/// file name as "myfile.gif+" or "myfile.gif+NN", where NN*10ms
 is delay
/// between the subimages' display. If NN is omitted the delay b
etween
/// subimages is zero. Each picture is added in the animation th
anks to a loop
/// similar to the following one:

for (int i=0; i<10; ++i) {
  // fill canvas for context i
  // ...
  c1.Print("file.gif+5");  // print canvas to GIF file with 50ms
 delays
 }// end loop

/// The delay between each frame must be specified in each Print
() statement.
/// If the file "myfile.gif" already exists, the new frame are a
ppended at
/// the end of the file. To avoid this, delete it first with gSy
stem->Unlink(myfile.gif);
/// If you want the gif file to repeat or loop forever, check TA
SImage::WriteImage documentation
```

example

TPaveStats

继承 TPaveText

class

```
   TPaveStats();
   TPaveStats(Double_t x1, Double_t y1,Double_t x2 ,Double_t y2,
 Option_t *option="br");/// TPaveStats normal constructor.
   virtual ~TPaveStats();
   virtual TBox    *AddBox(Double_t , Double_t , Double_t , Doub
le_t) {return 0;}
   virtual TLine   *AddLine(Double_t , Double_t , Double_t, Doub
le_t) {return 0;}
   virtual void     DeleteText() { }
   virtual void     EditText() { }
   virtual const char  *GetFitFormat()  const {return fFitFormat
.Data();}
   virtual const char  *GetStatFormat() const {return fStatForma
t.Data();}
   Int_t            GetOptFit() const;/// Return the fit option.
   Int_t            GetOptStat() const;/// Return the stat optio
n.
   TObject         *GetParent() const {return fParent;}
   virtual void     Paint(Option_t *option="");/// Paint the pav
e stat.
   virtual void     InsertText(const char *) { }
   virtual void     InsertLine() { }
   virtual void     ReadFile(const char *, Option_t *, Int_t, In
t_t) { }
   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
/// Save primitive as a C++ statement(s) on output stream out.

   virtual void     SaveStyle(); // *MENU*
/// Save This TPaveStats options in current style.
```

```
   virtual void     SetAllWith(const char *, Option_t *, Double_
t) { }
   virtual void     SetMargin(Float_t) { }
   virtual void     SetFitFormat(const char *format="5.4g");
// *MENU*
/// Change (i.e. set) the format for printing fit parameters in
statistics box.

   virtual void     SetStatFormat(const char *format="6.4g");
// *MENU*
/// Change (i.e. set) the format for printing statistics.

   void             SetOptFit(Int_t fit=1);
// *MENU*
/// Set the fit option.

   void             SetOptStat(Int_t stat=1);
// *MENU*
/// Set the stat option.

   void             SetParent(TObject*obj) {fParent = obj;}
   virtual void     UseCurrentStyle();
/// Replace current attributes by current style.
```

code

```
//设置TPave参数
TPaveStats *ps2 = (TPaveStats*)h2->GetListOfFunctions()->FindObj
ect("stats");
ps2->SetX1NDC(0.65); ps2->SetX2NDC(0.85);//设置位置
ps2->SetTextColor(kRed);//设置颜色
```

```
// When a histogram is painted, a TPaveStats object is created a
nd added
// to the list of functions of the histogram. If a TPaveStats ob
ject
// already exists in the histogram list of functions, the existi
```

```
ng object is just
// updated with the current histogram parameters.

// Once a histogram is painted, the statistics box can be access
ed using
// h->FindObject("stats"). In the command line it is enough to d
o:

Root > h->Draw()
Root > TPaveStats *st = (TPaveStats*)h->FindObject("stats")

// because after h->Draw() the histogram is automatically painte
d. But
// in a script file the painting should be forced using gPad->Up
date()
// in order to make sure the statistics box is created:

h->Draw();
gPad->Update();
TPaveStats *st = (TPaveStats*)h->FindObject("stats");

// Without gPad->Update() the line h->FindObject("stats")
// returns a null pointer.

// When a histogram is drawn with the option "SAME", the statist
ics box
// is not drawn. To force the statistics box drawing with the op
tion
// "SAME", the option "SAMES" must be used.
// If the new statistics box hides the previous statistics box,
one can change
// its position with these lines ("h" being the pointer to the h
istogram):

Root > TPaveStats *st = (TPaveStats*)h->FindObject("stats")
Root > st->SetX1NDC(newx1); //new x start position
Root > st->SetX2NDC(newx2); //new x end position

// To change the type of information for an histogram with an ex
isting
```

```
// TPaveStats one should do:

st->SetOptStat(mode);

// Where "mode" has the same meaning than when calling
// gStyle->SetOptStat(mode)` (see above).

// One can delete the statistics box for a histogram TH1* h with:


h->SetStats(0)

// and activate it again with:

h->SetStats(1).

// The type of information about fit parameters printed in the h
istogram statistics
// box can be selected via the parameter mode. The parameter mod
e can be
// = pcev  (default = 0111)

//      p = 1;  print Probability
//      c = 1;  print Chisquare/Number of degrees of freedom
//      e = 1;  print errors (if e=1, v must be 1)
//      v = 1;  print name/values of parameters

gStyle->SetOptFit(1011);

// print fit probability, parameter names/values and errors.

//  1. When "v" = 1 is specified, only the non-fixed parameters
are shown.
//  2. When "v" = 2 all parameters are shown.

// Note: gStyle->SetOptFit(1) means "default value", so it is eq
uivalent
// to gStyle->SetOptFit(111)
```

example

TPolyMarker

TPaveText

继承 public TPave, TAttText

A Pave with several lines of text.

class

```
   TPaveText();
   TPaveText(Double_t x1, Double_t y1,Double_t x2 ,Double_t y2,
Option_t *option="br");
   // PaveText normal constructor.
   // A PaveText is a Pave with several lines of text
   //  option = "TR" Top and Right shadows are drawn.
   //  option = "TL" Top and Left shadows are drawn.
   //  option = "BR" Bottom and Right shadows are drawn.
   //  option = "BL" Bottom and Left shadows are drawn.
   //  If none of these four above options is specified the defa
ult the
   //  option "BR" will be used to draw the border. To produces
a pave
   //  without any border it is enough to specify the option "NB
" (no border).
   //  option = "NDC" x1,y1,x2,y2 are given in NDC
   //  option = "ARC" corners are rounded
   // In case of option "ARC", the corner radius is specified
   // via TPave::SetCornerRadius(rad) where rad is given in perc
ent
   // of the pave height (default value is 0.2).
   // The individual text items are entered via AddText
   // By default, text items inherits from the default pavetext
AttText.
   // A title can be added later to this pavetext via TPaveText:
:SetLabel.

   TPaveText(const TPaveText &pavetext);
// pavetext copy constructor.
```

```
   virtual ~TPaveText();
   TPaveText& operator=(const TPaveText&);

   virtual TBox    *AddBox(Double_t x1, Double_t y1, Double_t x2
, Double_t y2);
// Add a new graphics box to this pavetext.

   virtual TLine   *AddLine(Double_t x1=0, Double_t y1=0, Double
_t x2=0, Double_t y2=0);
// Add a new graphics line to this pavetext.

   virtual TText   *AddText(Double_t x1, Double_t y1, const char
 *label);
// Add a new Text line to this pavetext at given coordinates.

   virtual TText   *AddText(const char *label);
// Add a new Text line to this pavetext.

   virtual void     Clear(Option_t *option="");  // *MENU*
// Clear all lines in this pavetext.

   virtual void     DeleteText(); // *MENU*
// Delete text at the mouse position.

   virtual void     Draw(Option_t *option="");
// Draw this pavetext with its current attributes.

   virtual void     DrawFile(const char *filename, Option_t *opt
ion="");
// Draw lines in filename in this pavetext.

   virtual void     EditText(); // *MENU*
// Edit text at the mouse position.

   const char      *GetLabel() const {return fLabel.Data();}
   virtual TText   *GetLine(Int_t number) const;
// Get Pointer to line number in this pavetext.

   virtual TText   *GetLineWith(const char *text) const;
```

```
// Get Pointer to first containing string text in this pavetext.

   virtual TList   *GetListOfLines() const {return fLines;}
   Float_t          GetMargin() const {return fMargin;}
   virtual TObject *GetObject(Double_t &ymouse, Double_t &yobj)
const;
// Get object pointed by the mouse in this pavetext.

   virtual Int_t    GetSize() const;
//  return number of text lines (ignoring Tlines, etc)

   virtual void     InsertLine(); // *MENU*
// Add a new lineine at the mouse position.

   virtual void     InsertText(const char *label); // *MENU*
// Add a new Text line at the mouse position.

   virtual void     Paint(Option_t *option="");
// Paint this pavetext with its current attributes.

   virtual void     PaintPrimitives(Int_t mode);
// Paint list of primitives in this pavetext.

   virtual void     Print(Option_t *option="") const;
   virtual void     ReadFile(const char *filename, Option_t *opt
ion="", Int_t nlines=50, Int_t fromline=0); // *MENU*
   // Read lines of filename in this pavetext.
   //  Read from line number fromline a total of nlines
   //  Note that this function changes the default text alignmen
t to left/center

   virtual void     SaveLines(std::ostream &out, const char *nam
e);
// Save lines of this pavetext as C++ statements on output strea
m out

   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
// Save primitive as a C++ statement(s) on output stream out
```

```
   virtual void     SetAllWith(const char *text, Option_t *optio
n, Double_t value); // *MENU*
   // Set attribute option for all lines containing string text.
   // Possible options are all the AttText attributes
   //       Align, Color, Font, Size and Angle

   virtual void     SetLabel(const char *label) {fLabel = label;
} // *MENU*
   virtual void     SetMargin(Float_t margin=0.05) {fMargin=marg
in;} // *MENU*
   virtual void     UseCurrentStyle();
```

code

example

```
TPaveText *pt = new TPaveText(0.6,0.85,0.98,0.98,"brNDC");//添加
图片的注释。
pt->SetFillColor(18);
pt->SetTextAlign(12);
pt->AddText("Use the axis Context Menu LabelsOption");
pt->AddText(" \"a\"   to sort by alphabetic order");
pt->AddText(" \">\"   to sort by decreasing vakues");
pt->AddText(" \"<\"   to sort by increasing vakues");
pt->Draw();
```

TPolyMarker3D

TProfile

TProfile2D

TProfile3D

TPServerSocket

TPSocket

TRandom

class

继承 **TNamed**

默认随机种子 65539

Simple Random number generator (periodicity = 10**9)

```
virtual  Int_t    Binomial(Int_t ntot, Double_t prob);//二项分布
virtual  Double_t BreitWigner(Double_t mean=0, Double_t gamma=1);
//Brei-Wigner分布
virtual  void     Circle(Double_t &x, Double_t &y, Double_t r);
virtual  Double_t Exp(Double_t tau);//指数分布
virtual  Double_t Gaus(Double_t mean=0, Double_t sigma=1);//高斯
分布
virtual  UInt_t   GetSeed() const {return fSeed;}//获得随机种子
virtual  UInt_t   Integer(UInt_t imax);//（0，imax-1）随机整数
virtual  Double_t Landau(Double_t mean=0, Double_t sigma=1);//La
ndau分布
virtual  Int_t    Poisson(Double_t mean);//泊松分布（返回int）
virtual  Double_t PoissonD(Double_t mean);//泊松分布（返回double）
virtual  void     Rannor(Float_t &a, Float_t &b);//Return 2 numb
ers distributed following a gaussian with mean=0 and sigma=1.
virtual  void     Rannor(Double_t &a, Double_t &b);//Return 2 nu
mbers distributed following a gaussian with mean=0 and sigma=1.
virtual  void     ReadRandom(const char *filename);//从root文件中
读取随机数产生器
virtual  void     SetSeed(UInt_t seed=0);//设置随机种子
virtual  Double_t Rndm(Int_t i=0);//（0，1]均匀分布
virtual  void     RndmArray(Int_t n, Float_t *array);
virtual  void     RndmArray(Int_t n, Double_t *array);
virtual  void     Sphere(Double_t &x, Double_t &y, Double_t &z,
Double_t r);//获得各向同性的抽样
virtual  Double_t Uniform(Double_t x1=1);
virtual  Double_t Uniform(Double_t x1, Double_t x2);//（x1，x2]均
匀分布
virtual  void     WriteRandom(const char *filename);//将该随机数产
生器存为root文件 Writes random generator status to filename.
```

code

```
#include "TRandom.h"
TRandom r;
r.Rndm();
r.Gaus();
r.Gaus(10,3);
```

```
// A TRandom object may be written to a Root file : as part of a
nother object or with its own key
gRandom->Write("Random");
```

example

```

```

TRandom1

class

继承 **TRandom**

Ranlux random number generator class (periodicity > 10**14)

```
   TRandom1();
   TRandom1(UInt_t seed, Int_t lux = 3 );
   TRandom1(Int_t rowIndex, Int_t colIndex, Int_t lux );
   virtual ~TRandom1();

   virtual  Int_t    GetLuxury() const {return fLuxury;}
                    // Get the current seed (first element of th
e table)
   virtual  UInt_t   GetSeed() const { return  UInt_t ( fFloatSe
edTable[0] /  fMantissaBit24 ) ; }
                    // Gets the current seed.
   const UInt_t     *GetTheSeeds() const {return fTheSeeds;}
                     // Gets the current array of seeds.
   static   void     GetTableSeeds(UInt_t* seeds, Int_t index);
                     // Gets back seed values stored in the tabl
e, given the index.
   virtual  Double_t Rndm(Int_t i=0);//（0，1]均匀分布
   virtual  void     RndmArray(Int_t size, Float_t *vect);
   virtual  void     RndmArray(Int_t size, Double_t *vect);
   virtual  void     SetSeed2(UInt_t seed, Int_t lux=3);
                     // Sets the state of the algorithm accordin
g to seed.
   virtual  void     SetSeeds(const UInt_t * seeds, Int_t lux=3)
;
                     // Sets the state of the algorithm accordin
g to the zero terminated
                     // array of seeds. Only the first seed is u
sed.
   virtual  void     SetSeed(UInt_t seed);
```

TRandom2

class

继承 **TRandom**

默认随机种子 1

TRandom2, is based on the Tausworthe generator of L'Ecuyer, and it has the advantage of being fast and using only 3 words (of 32 bits) for the state. The period is 10**26.

```
   virtual  Double_t Rndm(Int_t i=0);//Generate number in interv
al (0,1) : 0 and 1 are not included in the interval
   virtual  void     RndmArray(Int_t n, Float_t *array);
   virtual  void     RndmArray(Int_t n, Double_t *array);
   virtual  void     SetSeed(UInt_t seed=0);
```

TRandom3

class

继承 **TRandom**

默认随机种子 4357

```
// get the current seed (only first element of the seed table)
virtual  UInt_t    GetSeed() const { return fMt[0];}
virtual  Double_t  Rndm(Int_t i=0);//（0，1]均匀分布
virtual  void      RndmArray(Int_t n, Float_t *array);
virtual  void      RndmArray(Int_t n, Double_t *array);
virtual  void      SetSeed(UInt_t seed=0);
```

code

```
#include "TRandom1.h"          TRandom3 r(111);
#include "TRandom2.h"          TRandom3 r(0);
#include "TRandom3.h"          TRandom3 r(0);
```

example

TROOT

继承 TDirectory

The TROOT object is the entry point to the ROOT system.
The single instance of TROOT is accessible via the global gROOT.
Using the gROOT pointer one has access to basically every object created in a ROOT based program. The TROOT object is essentially a container of several lists pointing to the main ROOT objects.

class

```
   void              AddClass(TClass *cl);
   void              AddClassGenerator(TClassGenerator *gen);
   void              Browse(TBrowser *b);
   Bool_t            ClassSaved(TClass *cl);
   void              CloseFiles();
   void              EndOfProcessCleanups();
   virtual TObject  *FindObject(const char *name) const;
   virtual TObject  *FindObject(const TObject *obj) const;
   virtual TObject  *FindObjectAny(const char *name) const;
   virtual TObject  *FindObjectAnyFile(const char *name) const;
   TObject          *FindSpecialObject(const char *name, void *&
where);
   const char       *FindObjectClassName(const char *name) const
;
   const char       *FindObjectPathName(const TObject *obj) cons
t;
   TClass           *FindSTLClass(const char *name, Bool_t load,
 Bool_t silent = kFALSE) const;
   void              ForceStyle(Bool_t force = kTRUE) { fForceSt
yle = force; }
   Bool_t            FromPopUp() const { return fFromPopUp; }
   TPluginManager   *GetPluginManager() const { return fPluginMa
nager; }
   TApplication     *GetApplication() const { return fApplicatio
```

```
n; }
   TInterpreter     *GetInterpreter() const { return fInterprete
r; }
   TClass           *GetClass(const char *name, Bool_t load = kT
RUE, Bool_t silent = kFALSE) const;
   TClass           *GetClass(const type_info &typeinfo, Bool_t
load = kTRUE, Bool_t silent = kFALSE) const;
   TColor           *GetColor(Int_t color) const;
   const char       *GetConfigOptions() const { return fConfigOp
tions; }
   const char       *GetConfigFeatures() const { return fConfigF
eatures; }
   const char       *GetCutClassName() const { return fCutClassN
ame; }
   const char       *GetDefCanvasName() const { return fDefCanva
sName; }
   Bool_t            GetEditHistograms() const { return fEditHis
tograms; }
   Int_t             GetEditorMode() const { return fEditorMode;
 }
   Bool_t            GetForceStyle() const { return fForceStyle;
 }
   Int_t             GetBuiltDate() const { return fBuiltDate; }
   Int_t             GetBuiltTime() const { return fBuiltTime; }
   const char       *GetGitCommit() const { return fGitCommit; }
   const char       *GetGitBranch() const { return fGitBranch; }
   const char       *GetGitDate();
   Int_t             GetVersionDate() const { return fVersionDat
e; }
   Int_t             GetVersionTime() const { return fVersionTim
e; }
   Int_t             GetVersionInt() const { return fVersionInt;
 }
   Int_t             GetVersionCode() const { return fVersionCod
e; }
   const char       *GetVersion() const { return fVersion; }
   TCollection      *GetListOfClasses() const { return fClasses;
 }
   TSeqCollection   *GetListOfColors() const { return fColors; }
   TCollection      *GetListOfTypes(Bool_t load = kFALSE);
```

```
   TCollection      *GetListOfGlobals(Bool_t load = kFALSE);
   TCollection      *GetListOfGlobalFunctions(Bool_t load = kFAL
SE);
   TSeqCollection   *GetListOfClosedObjects() const { return fCl
osedObjects; }
   TSeqCollection   *GetListOfFiles() const       { return fFile
s; }
   TSeqCollection   *GetListOfMappedFiles() const { return fMapp
edFiles; }
   TSeqCollection   *GetListOfSockets() const     { return fSock
ets; }
   TSeqCollection   *GetListOfCanvases() const    { return fCanv
ases; }
   TSeqCollection   *GetListOfStyles() const      { return fStyl
es; }
   TCollection      *GetListOfFunctions() const   { return fFunc
tions; }
   TCollection      *GetListOfFunctionOverloads(const char* name
) const;
   TSeqCollection   *GetListOfGeometries() const  { return fGeom
etries; }
   TSeqCollection   *GetListOfBrowsers() const    { return fBrow
sers; }
   TSeqCollection   *GetListOfSpecials() const    { return fSpec
ials; }
   TSeqCollection   *GetListOfTasks() const       { return fTask
s; }
   TSeqCollection   *GetListOfCleanups() const    { return fClea
nups; }
   TSeqCollection   *GetListOfStreamerInfo() const { return fStr
eamerInfo; }
   TSeqCollection   *GetListOfMessageHandlers() const { return f
MessageHandlers; }
   TCollection      *GetListOfClassGenerators() const { return f
ClassGenerators; }
   TSeqCollection   *GetListOfSecContexts() const { return fSecC
ontexts; }
   TSeqCollection   *GetListOfProofs() const { return fProofs; }
   TSeqCollection   *GetClipboard() const { return fClipboard; }
   TSeqCollection   *GetListOfDataSets() const { return fDataSet
```

```
s; }
   TCollection      *GetListOfEnums(Bool_t load = kFALSE);
   TCollection      *GetListOfFunctionTemplates();
   TList            *GetListOfBrowsables() const { return fBrows
ables; }
   TDataType        *GetType(const char *name, Bool_t load = kFA
LSE) const;
   TFile            *GetFile() const { if (gDirectory != this) r
eturn gDirectory->GetFile(); else return 0;}
   TFile            *GetFile(const char *name) const;
   TFunctionTemplate*GetFunctionTemplate(const char *name);
   TStyle           *GetStyle(const char *name) const;
   TObject          *GetFunction(const char *name) const;
   TGlobal          *GetGlobal(const char *name, Bool_t load = k
FALSE) const;
   TGlobal          *GetGlobal(const TObject *obj, Bool_t load =
 kFALSE) const;
   TFunction        *GetGlobalFunction(const char *name, const c
har *params = 0, Bool_t load = kFALSE);
   TFunction        *GetGlobalFunctionWithPrototype(const char *
name, const char *proto = 0, Bool_t load = kFALSE);
   TObject          *GetGeometry(const char *name) const;
   const TObject    *GetSelectedPrimitive() const { return fPrim
itive; }
   TVirtualPad      *GetSelectedPad() const { return fSelectPad;
 }
   Int_t             GetNclasses() const { return fClasses->GetS
ize(); }
   Int_t             GetNtypes() const { return fTypes->GetSize(
); }
   TFolder          *GetRootFolder() const { return fRootFolder;
 }
   TProcessUUID     *GetUUIDs() const { return fUUIDs; }
   void              Idle(UInt_t idleTimeInSec, const char *comm
and = 0);
   Int_t             IgnoreInclude(const char *fname, const char
 *expandedfname);
   Bool_t            IsBatch() const { return fBatch; }
   Bool_t            IsExecutingMacro() const { return fExecutin
gMacro; }
```

```
   Bool_t            IsFolder() const { return kTRUE; }
   Bool_t            IsInterrupted() const { return fInterrupt;
}
   Bool_t            IsEscaped() const { return fEscape; }
   Bool_t            IsLineProcessing() const { return fLineIsPr
ocessing ? kTRUE : kFALSE; }
   Bool_t            IsProofServ() const { return fName == "proo
fserv" ? kTRUE : kFALSE; }
   Bool_t            IsRootFile(const char *filename) const;
   void              ls(Option_t *option = "") const;
   Int_t             LoadClass(const char *classname, const char
 *libname, Bool_t check = kFALSE);
   TClass           *LoadClass(const char *name, Bool_t silent =
 kFALSE) const;
   Int_t             LoadMacro(const char *filename, Int_t *erro
r = 0, Bool_t check = kFALSE);
   Long_t            Macro(const char *filename, Int_t *error =
0, Bool_t padUpdate = kTRUE);
   TCanvas          *MakeDefCanvas() const;
   void              Message(Int_t id, const TObject *obj);
   Bool_t            MustClean() const { return fMustClean; }
   Long_t            ProcessLine(const char *line, Int_t *error
= 0);
   Long_t            ProcessLineSync(const char *line, Int_t *er
ror = 0);
   Long_t            ProcessLineFast(const char *line, Int_t *er
ror = 0);
   Bool_t            ReadingObject() const;
   void              RefreshBrowsers();
   static void       RegisterModule(const char* modulename,
                                    const char** headers,
                                    const char** includePaths,
                                    const char* payLoadCode,
                                    const char* fwdDeclCode,
                                    void (*triggerFunc)(),
                                    const FwdDeclArgsToKeepColle
ction_t& fwdDeclsArgToSkip,
                                    const char** classesHeaders)
;
   void              RemoveClass(TClass *);
```

```
   void              Reset(Option_t *option="");
   void              SaveContext();
   void              SetApplication(TApplication *app) { fApplic
ation = app; }
   void              SetBatch(Bool_t batch = kTRUE) { fBatch = b
atch; }
   void              SetCutClassName(const char *name = "TCutG")
;
   void              SetDefCanvasName(const char *name = "c1") {
 fDefCanvasName = name; }
   void              SetEditHistograms(Bool_t flag = kTRUE) { fE
ditHistograms = flag; }
   void              SetEditorMode(const char *mode = "");
   void              SetExecutingMacro(Bool_t flag = kTRUE) { fE
xecutingMacro = flag; }
   void              SetFromPopUp(Bool_t flag = kTRUE) { fFromPo
pUp = flag; }
   void              SetInterrupt(Bool_t flag = kTRUE) { fInterr
upt = flag; }
   void              SetEscape(Bool_t flag = kTRUE) { fEscape =
flag; }
   void              SetLineIsProcessing() { fLineIsProcessing++
; }
   void              SetLineHasBeenProcessed() { if (fLineIsProc
essing) fLineIsProcessing--; }
   void              SetReadingObject(Bool_t flag = kTRUE);
   void              SetMustClean(Bool_t flag = kTRUE) { fMustCl
ean=flag; }
   void              SetSelectedPrimitive(const TObject *obj) {
fPrimitive = obj; }
   void              SetSelectedPad(TVirtualPad *pad) { fSelectP
ad = pad; }
   void              SetStyle(const char *stylename = "Default")
;
   void              Time(Int_t casetime=1) { fTimer = casetime;
 }
   Int_t             Timer() const { return fTimer; }

   //---- static functions
   static Int_t       DecreaseDirLevel();
```

```
    static Int_t       GetDirLevel();
    static const char *GetMacroPath();
    static void        SetMacroPath(const char *newpath);
    static Int_t       IncreaseDirLevel();
    static void        IndentLevel();
    static Bool_t      Initialized();
    static Bool_t      MemCheck();
    static void        SetDirLevel(Int_t level = 0);
    static Int_t       ConvertVersionCode2Int(Int_t code);
    static Int_t       ConvertVersionInt2Code(Int_t v);
    static Int_t       RootVersionCode();
    static const char**&GetExtraInterpreterArgs();
    static const char *GetTutorialsDir();
```

code

```
// The following lists are accessible from gROOT object:
gROOT->GetListOfClasses
gROOT->GetListOfColors
gROOT->GetListOfTypes
gROOT->GetListOfGlobals
gROOT->GetListOfGlobalFunctions
gROOT->GetListOfFiles
gROOT->GetListOfMappedFiles
gROOT->GetListOfSockets
gROOT->GetListOfSecContexts
gROOT->GetListOfCanvases
gROOT->GetListOfStyles
gROOT->GetListOfFunctions
gROOT->GetListOfSpecials (for example graphical cuts)
gROOT->GetListOfGeometries
gROOT->GetListOfBrowsers
gROOT->GetListOfCleanups
gROOT->GetListOfMessageHandlers
```

```
//  The ROOT object must be created as a static object. An examp
le
//  of a main program creating an interactive version is shown b
elow:

#include "TRint.h"

int main(int argc, char **argv)
{
   TRint *theApp = new TRint("ROOT example", &argc, argv);

   // Init Intrinsics, build all windows, and enter event loop
   theApp->Run();

   return(0);
}
```

TServerSocket

TSocket

TSpectrum

class

```
TSpectrum();
TSpectrum(Int_t maxpositions, Double_t resolution=1);
virtual ~TSpectrum();
virtual TH1        *Background(const TH1 *hist,Int_t niter=20, O
ption_t *option="");
TH1                *GetHistogram() const {return fHistogram;}//
暂未实施
Int_t               GetNPeaks() const {return fNPeaks;}
Double_t            *GetPositionX() const {return fPositionX;}
Double_t            *GetPositionY() const {return fPositionY;}
virtual void        Print(Option_t *option="") const;
virtual Int_t       Search(const TH1 *hist, Double_t sigma=2, Op
tion_t *option="", Double_t threshold=0.05);//option 可选
static void         SetAverageWindow(Int_t w=3);   //set average
 window
static void         SetDeconIterations(Int_t n=3); //set max num
ber of decon iterations
void                SetResolution(Double_t resolution=1);

//new functions January 2006
const char         *Background(Double_t *spectrum, Int_t ssize,I
nt_t numberIterations,Int_t direction, Int_t filterOrder,bool sm
oothing,Int_t smoothWindow,bool compton);//实际计算background函数
const char         *SmoothMarkov(Double_t *source, Int_t ssize,
Int_t averWindow);
const char         *Deconvolution(Double_t *source, const Double
_t *response,Int_t ssize, Int_t numberIterations,Int_t numberRep
etitions, Double_t boost );
const char         *DeconvolutionRL(Double_t *source, const Doub
le_t *response,Int_t ssize, Int_t numberIterations,Int_t numberR
epetitions, Double_t boost );
const char         *Unfolding(Double_t *source,const Double_t **
respMatrix,Int_t ssizex, Int_t ssizey,Int_t numberIterations,Int
```

```
_t numberRepetitions, Double_t boost);
Int_t                SearchHighRes(Double_t *source,Double_t *des
tVector, Int_t ssize,Double_t sigma, Double_t threshold,bool bac
kgroundRemove,Int_t deconIterations,bool markov, Int_t averWindo
w);//实际寻峰函数
Int_t                Search1HighRes(Double_t *source,Double_t *de
stVector, Int_t ssize,Double_t sigma, Double_t threshold,bool ba
ckgroundRemove,Int_t deconIterations,bool markov, Int_t averWind
ow);

static Int_t         StaticSearch(const TH1 *hist, Double_t sigma=
2, Option_t *option="goff", Double_t threshold=0.05);
static TH1          *StaticBackground(const TH1 *hist,Int_t niter=
20, Option_t *option="");
```

```
virtual TH1         *Background(const TH1 *hist,Int_t niter=20, O
ption_t *option="");

enum {
  kBackOrder2 =0,
  kBackOrder4 =1,
  kBackOrder6 =2,
  kBackOrder8 =3,
  kBackIncreasingWindow =0,
  kBackDecreasingWindow =1,
  kBackSmoothing3 =3,
  kBackSmoothing5 =5,
  kBackSmoothing7 =7,
  kBackSmoothing9 =9,
  kBackSmoothing11 =11,
  kBackSmoothing13 =13,
  kBackSmoothing15 =15
};

// One-dimensional background estimation function.
// This function calculates the background spectrum in the input
 histogram h.The background is returned as a histogram.
```

```
// Function parameters:

// h: input 1-d histogram
// numberIterations, (default value = 20).Increasing numberItera
tions make the result smoother and lower.
// option: may contain one of the following options:

// to set the direction parameter
// "BackIncreasingWindow". By default the direction is BackDecre
asingWindow
// filterOrder-order of clipping filter,  (default "BackOrder2")
// -possible values= "BackOrder4"
// "BackOrder6"
// "BackOrder8"
// "nosmoothing"- if selected, the background is not smoothed
// By default the background is smoothed.
// smoothWindow-width of smoothing window, (default is "BackSmoo
thing3")
// -possible values= "BackSmoothing5"
// "BackSmoothing7"
// "BackSmoothing9"
// "BackSmoothing11"
// "BackSmoothing13"
// "BackSmoothing15"
// "Compton" if selected the estimation of Compton edge
// will be included.
// "same" : if this option is specified, the resulting background

// histogram is superimposed on the picture in the current pad.

// NOTE that the background is only evaluated in the current ran
ge of h.
// ie, if h has a bin range (set via h->GetXaxis()->SetRange(bin
min,binmax),
// the returned histogram will be created with the same number o
f bins
// as the input histogram h, but only bins from binmin to binmax
 will be filled
// with the estimated background.
```

```
virtual Int_t       Search(const TH1 *hist, Double_t sigma=2, Op
tion_t *option="", Double_t threshold=0.05);

// One-dimensional peak search function
// This function searches for peaks in source spectrum in hin.Th
e number of found peaks and their positions are written into the
 members fNpeaks and fPositionX. The search is performed in the
current histogram range.

// Function parameters:
// hin:       pointer to the histogram of source spectrum
// sigma:   sigma of searched peaks, for details we refer to man
ual
// threshold: (default=0.05)  peaks with amplitude less than
// threshold*highest_peak are discarded.  0<threshold<1

// By default, the background is removed before deconvolution.
// Specify the option "nobackground" to not remove the backgroun
d.

// By default the "Markov" chain algorithm is used.
// Specify the option "noMarkov" to disable this algorithm
// Note that by default the source spectrum is replaced by a new
 spectrum

// By default a polymarker object is created and added to the li
st of
// functions of the histogram. The histogram is drawn with the s
pecified
// option and the polymarker object drawn on top of the histogra
m.
// The polymarker coordinates correspond to the npeaks peaks fou
nd in
// the histogram.

// A pointer to the polymarker object can be retrieved later via:


// TList *functions = hin->GetListOfFunctions();
```

```
// TPolyMarker *pm = (TPolyMarker*)functions->FindObject("TPolyM
arker");

// Specify the option "goff" to disable the storage and drawing
of the
// polymarker.
// To disable the final drawing of the histogram with the search
 results (in case
// you want to draw it yourself) specify "nodraw" in the options
 parameter.
```

```
const char          *Background(Double_t *spectrum, Int_t ssize,I
nt_t numberIterations,Int_t direction, Int_t filterOrder,bool sm
oothing,Int_t smoothWindow,bool compton);
// This function calculates background spectrum from source spec
trum.
// The result is placed in the vector pointed by spe1945ctrum po
inter.
// The goal is to separate the useful information (peaks) from u
seless
// information (background).

// method is based on Sensitive Nonlinear Iterative Peak (SNIP)
clipping algorithm.
// new value in the channel "i" is calculated

// where p = 1, 2, ..., numberIterations. In fact it represents
second order
// difference filter (-1,2,-1).

// One can also change the
// direction of the change of the clipping window, the order of
the clipping
// filter, to include smoothing, to set width of smoothing windo
w and to include
// the estimation of Compton edges. On successful completion it
returns 0. On
// error it returns pointer to the string describing error.
```

```
// Parameters:
// spectrum: pointer to the vector of source spectrum
// ssize: length of the spectrum vector
// numberIterations: maximal width of clipping window,
// direction:  direction of change of clipping window.Possible v
alues: kBackIncreasingWindow, kBackDecreasingWindow
// filterOrder: order of clipping filter.Possible values: kBackO
rder2, kBackOrder4, kBackOrder6, kBackOrder8
// smoothing: logical variable whether the smoothing operation i
n the estimation of background will be included.Possible values:
 kFALSE, kTRUE
// smoothWindow: width of smoothing window.Possible values: kBac
kSmoothing3, kBackSmoothing5, kBackSmoothing7,kBackSmoothing9, k
BackSmoothing11, kBackSmoothing13, kBackSmoothing15.
// compton: logical variable whether the estimation of Compton e
dge will be included. Possible values: kFALSE, kTRUE.
```

code

```
Int_t npeaks=10;
TH1F *h = new TH1F("h","test",500,0,1000);
TSpectrum *s = new TSpectrum(2*npeaks);
Int_t nfound = s->Search(h,2,"",0.1);
double *number=s->GetPositionX();
for (int i = 0; i<nfound; ++i)
  {
    cout<<number[i]<<endl;
  }
cout<<"!!!:"<<s->GetNPeaks()<<endl;
```

example

以下例子年代久远，仅供参考，不能运行：

```
#include <TSpectrum>
void Background_incr() {
   Int_t i;
   Double_t nbins = 256;
   Double_t xmin  = 0;
   Double_t xmax  = nbins;
   Double_t * source = new Double_t[nbins];
   TH1F *back = new TH1F("back","",nbins,xmin,xmax);
   TH1F *d = new TH1F("d","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   back=(TH1F*) f->Get("back1;1");
   TCanvas *Background = gROOT->GetListOfCanvases()->FindObject(
"Background");
   if (!Background) Background =
     new TCanvas("Background",
                 "Estimation of background with increasing windo
w",
                 10,10,1000,700);
   back->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=back->GetBinContent(i +
1);
   s->Background(source,nbins,6,kBackIncreasingWindow,kBackOrder
2,kFALSE,
                 kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d->SetBinContent(i + 1,source[i])
;
   d->SetLineColor(kRed);
   d->Draw("SAME L");
}
```

```
#include <TSpectrum>
void Background_decr() {
   Int_t i;
   Double_t nbins = 256;
   Double_t xmin  = 0;
   Double_t xmax  = nbins;
   Double_t * source = new Double_t[nbins];
   TH1F *back = new TH1F("back","",nbins,xmin,xmax);
   TH1F *d = new TH1F("d","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   back=(TH1F*) f->Get("back1;1");
   TCanvas *Background = gROOT->GetListOfCanvases()->FindObject(
"Background");
   if (!Background) Background =
     new TCanvas("Background","Estimation of background with dec
reasing window",
                 10,10,1000,700);
   back->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=back->GetBinContent(i +
1);
   s->Background(source,nbins,6,kBackDecreasingWindow,kBackOrder
2,kFALSE,
                 kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d->SetBinContent(i + 1,source[i])
;
   d->SetLineColor(kRed);
   d->Draw("SAME L");
}
```

```
#include <TSpectrum>
void Background_width() {
   Int_t i;
   Double_t nbins = 256;
   Double_t xmin  = 0;
   Double_t xmax  = nbins;
   Double_t * source = new Double_t[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
```

```
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TH1F *d2 = new TH1F("d2","",nbins,xmin,xmax);
   TH1F *d3 = new TH1F("d3","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back1;1");
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject(
"background");
   if (!background) background = new TCanvas("background",
   "Influence of clipping window width on the estimated backgrou
nd",
   10,10,1000,700);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,4,kBackDecreasingWindow,kBackOrder
2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]
);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,6,kBackDecreasingWindow,kBackOrder
2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d2->SetBinContent(i + 1,source[i]
);
   d2->SetLineColor(kBlue);
   d2->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,8,kBackDecreasingWindow,kBackOrder
2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d3->SetBinContent(i + 1,source[i]
);
   d3->SetLineColor(kGreen);
   d3->Draw("SAME L");
```

```
}
```

```
#include <TSpectrum>
void Background_width2() {
   Int_t i;
   Double_t nbins = 4096;
   Double_t xmin  = 0;
   Double_t xmax  = 4096;
   Double_t * source = new Double_t[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TH1F *d2 = new TH1F("d2","",nbins,xmin,xmax);
   TH1F *d3 = new TH1F("d3","",nbins,xmin,xmax);
   TH1F *d4 = new TH1F("d4","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back2;1");
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject(
"background");
   if (!background) background = new TCanvas("background",
   "Influence of clipping window width on the estimated backgrou
nd",
   10,10,1000,700);
   h->SetAxisRange(0,1000);
   h->SetMaximum(20000);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,10,kBackDecreasingWindow,kBackOrde
r2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]
);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,20,kBackDecreasingWindow,kBackOrde
r2,kFALSE,
```

```
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d2->SetBinContent(i + 1,source[i]
);
   d2->SetLineColor(kBlue);
   d2->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,30,kBackDecreasingWindow,kBackOrde
r2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d3->SetBinContent(i + 1,source[i]
);
   d3->SetLineColor(kGreen);
   d3->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,10,kBackDecreasingWindow,kBackOrde
r2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d4->SetBinContent(i + 1,source[i]
);
   d4->SetLineColor(kMagenta);
   d4->Draw("SAME L");
}
```

```
#include <TSpectrum>
void Background_order() {
   Int_t i;
   Double_t nbins = 4096;
   Double_t xmin  = 0;
   Double_t xmax  = 4096;
   Double_t * source = new Double_t[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TH1F *d2 = new TH1F("d2","",nbins,xmin,xmax);
   TH1F *d3 = new TH1F("d3","",nbins,xmin,xmax);
   TH1F *d4 = new TH1F("d4","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back2;1");
```

```
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject(
"background");
   if (!background) background = new TCanvas("background",
   "Influence of clipping filter difference order on the estimat
ed background",
   10,10,1000,700);
   h->SetAxisRange(1220,1460);
   h->SetMaximum(11000);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,40,kBackDecreasingWindow,kBackOrde
r2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]
);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,40,kBackDecreasingWindow,kBackOrde
r4,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d2->SetBinContent(i + 1,source[i]
);
   d2->SetLineColor(kBlue);
   d2->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,40,kBackDecreasingWindow,kBackOrde
r6,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d3->SetBinContent(i + 1,source[i]
);
   d3->SetLineColor(kGreen);
   d3->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,40,kBackDecreasingWindow,kBackOrde
```

```
r8,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d4->SetBinContent(i + 1,source[i]
);
   d4->SetLineColor(kMagenta);
   d4->Draw("SAME L");
}
```

```
#include <TSpectrum>
void Background_smooth() {
   Int_t i;
   Double_t nbins = 4096;
   Double_t xmin  = 0;
   Double_t xmax  = nbins;
   Double_t * source = new Double_t[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TH1F *d2 = new TH1F("d2","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back4;1");
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject(
"background");
   if (!background) background = new TCanvas("background",
   "Estimation of background with noise",10,10,1000,700);
   h->SetAxisRange(3460,3830);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,6,kBackDecreasingWindow,kBackOrder
2,kFALSE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]
);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,6,kBackDecreasingWindow,kBackOrder
2,kTRUE,
   kBackSmoothing3,kFALSE);
   for (i = 0; i < nbins; i++) d2->SetBinContent(i + 1,source[i]
);
   d2->SetLineColor(kBlue);
   d2->Draw("SAME L");
}
```

```
#include <TSpectrum>
void Background_compton() {
   Int_t i;
   Double_t nbins = 512;
   Double_t xmin  = 0;
   Double_t xmax  = nbins;
   Double_t * source = new Double_t[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back3;1");
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject(
"background");
   if (!background) background = new TCanvas("background",
   "Estimation of background with Compton edges under peaks",10,
10,1000,700);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1)
;
   s->Background(source,nbins,10,kBackDecreasingWindow,kBackOrde
r8,kTRUE,
   kBackSmoothing5,,kTRUE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]
);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
}
```

TSpectrum2

TSpectrum2Fit

TSpectrum2Painter

TSpectrum2Transform

TSpectrum3

TSpectrumFit

TSpectrumTransform

TSpline

直接使用的插值类：TSpline3、TSpline5

TSpline3 继承于 TSpline TSpline5 继承于 TSpline

抽象基类 TSpline 继承于 TNamed, TAttLine, TAttFill, TAttMarker

TSplinePoly 继承于 TObject TSplinePoly3 继承于 TSplinePoly TSplinePoly5 继承于 TSplinePoly

class

TSpline3

```
public:
   TSpline3() : TSpline() , fPoly(0), fValBeg(0), fValEnd(0),
      fBegCond(-1), fEndCond(-1) {}
   TSpline3(const char *title,
            Double_t x[], Double_t y[], Int_t n, const char *opt=
0,
            Double_t valbeg=0, Double_t valend=0);
   TSpline3(const char *title,
            Double_t xmin, Double_t xmax,
            Double_t y[], Int_t n, const char *opt=0,
            Double_t valbeg=0, Double_t valend=0);
   TSpline3(const char *title,
            Double_t x[], const TF1 *func, Int_t n, const char *
opt=0,
            Double_t valbeg=0, Double_t valend=0);
   TSpline3(const char *title,
            Double_t xmin, Double_t xmax,
            const TF1 *func, Int_t n, const char *opt=0,
            Double_t valbeg=0, Double_t valend=0);
   TSpline3(const char *title,
            const TGraph *g, const char *opt=0,
            Double_t valbeg=0, Double_t valend=0);
```

```
   TSpline3(const TH1 *h, const char *opt=0,
            Double_t valbeg=0, Double_t valend=0);
   TSpline3(const TSpline3&);
   TSpline3& operator=(const TSpline3&);
   Int_t    FindX(Double_t x) const;
   Double_t Eval(Double_t x) const;
   Double_t Derivative(Double_t x) const;
   virtual ~TSpline3() {if (fPoly) delete [] fPoly;}
   void GetCoeff(Int_t i, Double_t &x, Double_t &y, Double_t &b,
                 Double_t &c, Double_t &d) {x=fPoly[i].X();y=fPo
ly[i].Y();
                  b=fPoly[i].B();c=fPoly[i].C();d=fPoly[i].D();}
   void GetKnot(Int_t i, Double_t &x, Double_t &y) const
      {x=fPoly[i].X(); y=fPoly[i].Y();}
   virtual  void     SaveAs(const char *filename,Option_t *optio
n="") const;
   virtual  void     SavePrimitive(std::ostream &out, Option_t *
option = "");
   virtual  void     SetPoint(Int_t i, Double_t x, Double_t y);
   virtual  void     SetPointCoeff(Int_t i, Double_t b, Double_t
 c, Double_t d);
   static void Test();
```

TSpline5

```
public:
   TSpline5() : TSpline() , fPoly(0) {}
   TSpline5(const char *title,
            Double_t x[], Double_t y[], Int_t n,
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const char *title,
            Double_t xmin, Double_t xmax,
            Double_t y[], Int_t n,
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const char *title,
            Double_t x[], const TF1 *func, Int_t n,
```

```
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const char *title,
            Double_t xmin, Double_t xmax,
            const TF1 *func, Int_t n,
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const char *title,
            const TGraph *g,
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const TH1 *h,
            const char *opt=0, Double_t b1=0, Double_t e1=0,
            Double_t b2=0, Double_t e2=0);
   TSpline5(const TSpline5&);
   TSpline5& operator=(const TSpline5&);
   Int_t    FindX(Double_t x) const;
   Double_t Eval(Double_t x) const;
   Double_t Derivative(Double_t x) const;
   virtual ~TSpline5() {if (fPoly) delete [] fPoly;}
   void GetCoeff(Int_t i, Double_t &x, Double_t &y, Double_t &b,
                 Double_t &c, Double_t &d, Double_t &e, Double_t
 &f)
      {x=fPoly[i].X();y=fPoly[i].Y();b=fPoly[i].B();
      c=fPoly[i].C();d=fPoly[i].D();
      e=fPoly[i].E();f=fPoly[i].F();}
   void GetKnot(Int_t i, Double_t &x, Double_t &y) const
      {x=fPoly[i].X(); y=fPoly[i].Y();}
   virtual  void     SaveAs(const char *filename,Option_t *optio
n="") const;
   virtual  void     SavePrimitive(std::ostream &out, Option_t *
option = "");
   virtual  void     SetPoint(Int_t i, Double_t x, Double_t y);
   virtual  void     SetPointCoeff(Int_t i, Double_t b, Double_t
 c, Double_t d,
                                   Double_t e, Double_t f);
   static void Test();
```

TSpline

```
public:
   TSpline() : fDelta(-1), fXmin(0), fXmax(0),
      fNp(0), fKstep(kFALSE), fHistogram(0), fGraph(0), fNpx(100
) {}
   TSpline(const char *title, Double_t delta, Double_t xmin,
      Double_t xmax, Int_t np, Bool_t step) :
      TNamed("Spline",title), TAttFill(0,1),
      fDelta(delta), fXmin(xmin),
      fXmax(xmax), fNp(np), fKstep(step),
      fHistogram(0), fGraph(0), fNpx(100) {}
   virtual ~TSpline();

   virtual void     GetKnot(Int_t i, Double_t &x, Double_t &y) c
onst =0;
   virtual Int_t    DistancetoPrimitive(Int_t px, Int_t py);
   virtual void     Draw(Option_t *option="");
   virtual void     ExecuteEvent(Int_t event, Int_t px, Int_t py)
;
   virtual Double_t GetDelta() const {return fDelta;}
   TH1F            *GetHistogram() const {return fHistogram;}
   virtual Int_t    GetNp()    const {return fNp;}
   virtual Int_t    GetNpx()   const {return fNpx;}
   virtual Double_t GetXmin()  const {return fXmin;}
   virtual Double_t GetXmax()  const {return fXmax;}
   virtual void     Paint(Option_t *option="");
   virtual Double_t Eval(Double_t x) const=0;
   virtual void     SaveAs(const char * /*filename*/,Option_t *
/*option*/) const {;}
   void             SetNpx(Int_t n) {fNpx=n;}
```

TSplinePoly

```
public:
   TSplinePoly() :
      fX(0), fY(0) {}
   TSplinePoly(Double_t x, Double_t y) :
      fX(x), fY(y) {}
   TSplinePoly(TSplinePoly const &other);
   TSplinePoly &operator=(TSplinePoly const &other);

   Double_t &X() {return fX;}
   Double_t &Y() {return fY;}
   void GetKnot(Double_t &x, Double_t &y) const {x=fX; y=fY;}

   virtual Double_t Eval(Double_t) const {return fY;}
```

TSplinePoly3

```
public:
   TSplinePoly3() :
      fB(0), fC(0), fD(0) {}
   TSplinePoly3(Double_t x, Double_t y, Double_t b, Double_t c,
Double_t d) :
      TSplinePoly(x,y), fB(b), fC(c), fD(d) {}
   TSplinePoly3(TSplinePoly3 const &other);
   TSplinePoly3 &operator=(TSplinePoly3 const &other);

   Double_t &B() {return fB;}
   Double_t &C() {return fC;}
   Double_t &D() {return fD;}
   Double_t Eval(Double_t x) const {
      Double_t dx=x-fX;
      return (fY+dx*(fB+dx*(fC+dx*fD)));
   }
   Double_t Derivative(Double_t x) const {
      Double_t dx=x-fX;
      return (fB+dx*(2*fC+3*fD*dx));
   }
```

TSplinePoly5

```
public:
   TSplinePoly5() :
      fB(0), fC(0), fD(0), fE(0), fF(0) {}
   TSplinePoly5(Double_t x, Double_t y, Double_t b, Double_t c,
      Double_t d, Double_t e, Double_t f) :
      TSplinePoly(x,y), fB(b), fC(c), fD(d), fE(e), fF(f) {}
   TSplinePoly5(TSplinePoly5 const &other);
   TSplinePoly5 &operator=(TSplinePoly5 const &other);

   Double_t &B() {return fB;}
   Double_t &C() {return fC;}
   Double_t &D() {return fD;}
   Double_t &E() {return fE;}
   Double_t &F() {return fF;}
   Double_t Eval(Double_t x) const {
      Double_t dx=x-fX;
      return (fY+dx*(fB+dx*(fC+dx*(fD+dx*(fE+dx*fF)))));
   }
   Double_t Derivative(Double_t x) const{
      Double_t dx=x-fX;
      return (fB+dx*(2*fC+dx*(3*fD+dx*(4*fE+dx*(5*fF)))));
   }
```

code

example

TStopwatch

```
void        Start(Bool_t reset = kTRUE);
void        Stop();
void        Continue();
Int_t       Counter() const { return fCounter; }
Double_t    RealTime();
void        Reset() { ResetCpuTime(); ResetRealTime(); }
void        ResetCpuTime(Double_t time = 0) { Stop();  fTotalCpu
Time = time; }
void        ResetRealTime(Double_t time = 0) { Stop(); fTotalRea
lTime = time; }
Double_t    CpuTime();
void        Print(Option_t *option="") const;
```

code

```
TStopwatch sw;
sw.Start();
printf("CPU: %8.3f\n",sw.CpuTime());
```

TString

Cannot be stored in a TCollection... use TObjString instead.

The underlying string is stored as a char* that can be accessed via TString::Data().
TString provides Short String Optimization (SSO) so that short strings (<15 on 64-bit and <11 on 32-bit) are contained in the TString internal data structure without the need for mallocing the required space.

class

```
   enum EStripType   { kLeading = 0x1, kTrailing = 0x2, kBoth =
0x3 };
   enum ECaseCompare { kExact, kIgnoreCase };
   static const Ssiz_t kNPOS = ::kNPOS;

   TString();                            // Null string
   explicit TString(Ssiz_t ic);          // Suggested capacity
   TString(const TString &s);            // Copy constructor
   TString(TString &&s);                 // Move constructor
   TString(const char *s);               // Copy to embedded null
   TString(const char *s, Ssiz_t n);     // Copy past any embedde
d nulls
   TString(const std::string &s);
   TString(char c);
   TString(char c, Ssiz_t s);
   TString(const std::string_view &sub);
   TString(const TSubString &sub);

   virtual ~TString();

   // ROOT I/O interface
   virtual void     FillBuffer(char *&buffer) const;
   virtual void     ReadBuffer(char *&buffer);
   virtual Int_t    Sizeof() const;
```

```
   static TString  *ReadString(TBuffer &b, const TClass *clReq);
   static void      WriteString(TBuffer &b, const TString *a);

   friend TBuffer &operator<<(TBuffer &b, const TString *obj);

   // C I/O interface
   Bool_t   Gets(FILE *fp, Bool_t chop=kTRUE);
   void     Puts(FILE *fp);

   // Type conversion
   operator const char*() const { return GetPointer(); }
   operator std::string_view() const { return std::string_view(G
etPointer(),Length()); }

   // Assignment
   TString    &operator=(char s);                // Replace stri
ng
   TString    &operator=(const char *s);
   TString    &operator=(const TString &s);
   TString    &operator=(const std::string &s);
   TString    &operator=(const std::string_view &s);
   TString    &operator=(const TSubString &s);
   TString    &operator+=(const char *s);        // Append strin
g
   TString    &operator+=(const TString &s);
   TString    &operator+=(char c);
   TString    &operator+=(Short_t i);
   TString    &operator+=(UShort_t i);
   TString    &operator+=(Int_t i);
   TString    &operator+=(UInt_t i);
   TString    &operator+=(Long_t i);
   TString    &operator+=(ULong_t i);
   TString    &operator+=(Float_t f);
   TString    &operator+=(Double_t f);
   TString    &operator+=(Long64_t i);
   TString    &operator+=(ULong64_t i);

   // Indexing operators
   char         &operator[](Ssiz_t i);         // Indexing with
```

```
bounds checking
   char         &operator()(Ssiz_t i);         // Indexing with
optional bounds checking
   char          operator[](Ssiz_t i) const;
   char          operator()(Ssiz_t i) const;
   TSubString    operator()(Ssiz_t start, Ssiz_t len) const;   /
/ Sub-string operator
   TSubString    operator()(const TRegexp &re) const;          /
/ Match the RE
   TSubString    operator()(const TRegexp &re, Ssiz_t start) con
st;
   TSubString    operator()(TPRegexp &re) const;               /
/ Match the Perl compatible Regular Expression
   TSubString    operator()(TPRegexp &re, Ssiz_t start) const;
   TSubString    SubString(const char *pat, Ssiz_t start = 0,
                           ECaseCompare cmp = kExact) const;

   // Non-static member functions
   TString     &Append(const char *cs);
   TString     &Append(const char *cs, Ssiz_t n);
   TString     &Append(const TString &s);
   TString     &Append(const TString &s, Ssiz_t n);
   TString     &Append(char c, Ssiz_t rep = 1);   // Append c re
p times
   Int_t        Atoi() const;
   Long64_t     Atoll() const;
   Double_t     Atof() const;
   Bool_t       BeginsWith(const char *s,      ECaseCompare cmp
= kExact) const;
   Bool_t       BeginsWith(const TString &pat, ECaseCompare cmp
= kExact) const;
   Ssiz_t       Capacity() const { return (IsLong() ? GetLongCap
() : kMinCap) - 1; }
   Ssiz_t       Capacity(Ssiz_t n);
   TString     &Chop();
   void         Clear();
   int          CompareTo(const char *cs,    ECaseCompare cmp =
kExact) const;
   int          CompareTo(const TString &st, ECaseCompare cmp =
kExact) const;
```

```
   Bool_t       Contains(const char *pat,    ECaseCompare cmp =
kExact) const;
   Bool_t       Contains(const TString &pat, ECaseCompare cmp =
kExact) const;
   Bool_t       Contains(const TRegexp &pat) const;
   Bool_t       Contains(TPRegexp &pat) const;
   Int_t        CountChar(Int_t c) const;
   TString      Copy() const;
   const char  *Data() const { return GetPointer(); }
   Bool_t       EndsWith(const char *pat, ECaseCompare cmp = kEx
act) const;
   Bool_t       EqualTo(const char *cs,    ECaseCompare cmp = kE
xact) const;
   Bool_t       EqualTo(const TString &st, ECaseCompare cmp = kE
xact) const;
   Ssiz_t       First(char c) const;
   Ssiz_t       First(const char *cs) const;
   void         Form(const char *fmt, ...)
#if defined(__GNUC__) && !defined(__CINT__)
   __attribute__((format(printf, 2, 3)))   /* 1 is the this poin
ter */
#endif
   ;
   UInt_t       Hash(ECaseCompare cmp = kExact) const;
   Ssiz_t       Index(const char *pat, Ssiz_t i = 0,
                      ECaseCompare cmp = kExact) const;
   Ssiz_t       Index(const TString &s, Ssiz_t i = 0,
                      ECaseCompare cmp = kExact) const;
   Ssiz_t       Index(const char *pat, Ssiz_t patlen, Ssiz_t i,
                      ECaseCompare cmp) const;
   Ssiz_t       Index(const TString &s, Ssiz_t patlen, Ssiz_t i,
                      ECaseCompare cmp) const;
   Ssiz_t       Index(const TRegexp &pat, Ssiz_t i = 0) const;
   Ssiz_t       Index(const TRegexp &pat, Ssiz_t *ext, Ssiz_t i
= 0) const;
   Ssiz_t       Index(TPRegexp &pat, Ssiz_t i = 0) const;
   Ssiz_t       Index(TPRegexp &pat, Ssiz_t *ext, Ssiz_t i = 0)
const;
   TString     &Insert(Ssiz_t pos, const char *s);
   TString     &Insert(Ssiz_t pos, const char *s, Ssiz_t extent)
```

```
;
   TString     &Insert(Ssiz_t pos, const TString &s);
   TString     &Insert(Ssiz_t pos, const TString &s, Ssiz_t exte
nt);
   Bool_t       IsAscii() const;
   Bool_t       IsAlpha() const;
   Bool_t       IsAlnum() const;
   Bool_t       IsDigit() const;
   Bool_t       IsFloat() const;
   Bool_t       IsHex() const;
   Bool_t       IsBin() const;
   Bool_t       IsOct() const;
   Bool_t       IsDec() const;
   Bool_t       IsInBaseN(Int_t base) const;
   Bool_t       IsNull() const         { return Length() == 0; }
   Bool_t       IsWhitespace() const   { return (Length() == Cou
ntChar(' ')); }
   Ssiz_t       Last(char c) const;
   Ssiz_t       Length() const         { return IsLong() ? GetLo
ngSize() : GetShortSize(); }
   Bool_t       MaybeRegexp() const;
   Bool_t       MaybeWildcard() const;
   TString      MD5() const;
   TString     &Prepend(const char *cs);     // Prepend a charac
ter string
   TString     &Prepend(const char *cs, Ssiz_t n);
   TString     &Prepend(const TString &s);
   TString     &Prepend(const TString &s, Ssiz_t n);
   TString     &Prepend(char c, Ssiz_t rep = 1);  // Prepend c r
ep times
   std::istream     &ReadFile(std::istream &str);      // Read t
o EOF or null character
   std::istream     &ReadLine(std::istream &str,
                          Bool_t skipWhite = kTRUE);   // Read to
 EOF or newline
   std::istream     &ReadString(std::istream &str);
// Read to EOF or null character
   std::istream     &ReadToDelim(std::istream &str, char delim =
 '\n'); // Read to EOF or delimitor
   std::istream     &ReadToken(std::istream &str);
```

```
  // Read separated by white space
   TString     &Remove(Ssiz_t pos);                      // Remov
e pos to end of string
   TString     &Remove(Ssiz_t pos, Ssiz_t n);            // Remov
e n chars starting at pos
   TString     &Remove(EStripType s, char c);            // Like
Strip() but changing string directly
   TString     &Replace(Ssiz_t pos, Ssiz_t n, const char *s);
   TString     &Replace(Ssiz_t pos, Ssiz_t n, const char *s, Ssi
z_t ns);
   TString     &Replace(Ssiz_t pos, Ssiz_t n, const TString &s);
   TString     &Replace(Ssiz_t pos, Ssiz_t n1, const TString &s,
 Ssiz_t n2);
   TString     &ReplaceAll(const TString &s1, const TString &s2)
; // Find&Replace all s1 with s2 if any
   TString     &ReplaceAll(const TString &s1, const char *s2);
  // Find&Replace all s1 with s2 if any
   TString     &ReplaceAll(const    char *s1, const TString &s2)
; // Find&Replace all s1 with s2 if any
   TString     &ReplaceAll(const char *s1, const char *s2);
  // Find&Replace all s1 with s2 if any
   TString     &ReplaceAll(const char *s1, Ssiz_t ls1, const cha
r *s2, Ssiz_t ls2);  // Find&Replace all s1 with s2 if any
   void         Resize(Ssiz_t n);                        // Trunc
ate or add blanks as necessary
   TSubString   Strip(EStripType s = kTrailing, char c = ' ') co
nst;
   TString     &Swap(TString &other); // Swap the contents of th
is and other without reallocation
   void         ToLower();                               // Chang
e self to lower-case
   void         ToUpper();                               // Chang
e self to upper-case
   TObjArray   *Tokenize(const TString &delim) const;
   Bool_t       Tokenize(TString &tok, Ssiz_t &from, const char
*delim = " ") const;

   // Static member functions
   static UInt_t  Hash(const void *txt, Int_t ntxt);    // Calcu
lates hash index from any char string.
```

```
   static Ssiz_t  InitialCapacity(Ssiz_t ic = 15);      // Initi
al allocation capacity
   static Ssiz_t  MaxWaste(Ssiz_t mw = 15);             // Max e
mpty space before reclaim
   static Ssiz_t  ResizeIncrement(Ssiz_t ri = 16);      // Resiz
ing increment
   static Ssiz_t  GetInitialCapacity();
   static Ssiz_t  GetResizeIncrement();
   static Ssiz_t  GetMaxWaste();
   static TString Itoa   (    Int_t value, Int_t base);  // Conv
erts int to string with respect to the base specified (2-36)
   static TString UItoa  (   UInt_t value, Int_t base);
   static TString LLtoa  ( Long64_t value, Int_t base);
   static TString ULLtoa (ULong64_t value, Int_t base);
   static TString BaseConvert(const TString& s_in, Int_t base_in
, Int_t base_out);  // Converts string from base base_in to base
 base_out (supported bases 2-36)
   static TString Format(const char *fmt, ...)
#if defined(__GNUC__) && !defined(__CINT__)
   __attribute__((format(printf, 1, 2)))
#endif
   ;
```

code

```
// Substring operations are provided by the TSubString class, wh
ich
// holds a reference to the original string and its data, along
with
// the offset and length of the substring. To retrieve the subst
ring
// as a TString, construct a TString from it, eg:
root [0] TString s("hello world")
root [1] TString s2( s(0,5) )
root [2] s2
(class TString)"hello"
```

example

TStyle

继承 TNamed, TAttLine, TAttFill, TAttMarker, TAttText

class

```
   virtual void     Browse(TBrowser *b);
   static  void     BuildStyles();
   virtual void     Copy(TObject &style) const;
   virtual void     cd();

   virtual Int_t    DistancetoPrimitive(Int_t px, Int_t py);
   Int_t            GetNdivisions(Option_t *axis="X") const;
   TAttText        *GetAttDate() {return &fAttDate;}
   Color_t          GetAxisColor(Option_t *axis="X") const;
   Color_t          GetLabelColor(Option_t *axis="X") const;
   Style_t          GetLabelFont(Option_t *axis="X") const;
   Float_t          GetLabelOffset(Option_t *axis="X") const;
   Float_t          GetLabelSize(Option_t *axis="X") const;
   Color_t          GetTitleColor(Option_t *axis="X") const;  //
return axis title color of pad title color
   Style_t          GetTitleFont(Option_t *axis="X") const;   //
return axis title font of pad title font
   Float_t          GetTitleOffset(Option_t *axis="X") const; //
return axis title offset
   Float_t          GetTitleSize(Option_t *axis="X") const;   //
return axis title size
   Float_t          GetTickLength(Option_t *axis="X") const;

   Float_t          GetBarOffset() const {return fBarOffset;}
   Float_t          GetBarWidth() const {return fBarWidth;}
   Int_t            GetDrawBorder() const {return fDrawBorder;}
   Float_t          GetEndErrorSize() const {return fEndErrorSiz
e;}
   Float_t          GetErrorX() const {return fErrorX;}
   Bool_t           GetCanvasPreferGL() const {return fCanvasPre
ferGL;}
```

```
   Color_t          GetCanvasColor() const {return fCanvasColor;
}
   Width_t          GetCanvasBorderSize() const {return fCanvasB
orderSize;}
   Int_t            GetCanvasBorderMode() const {return fCanvasB
orderMode;}
   Int_t            GetCanvasDefH() const      {return fCanvasDe
fH;}
   Int_t            GetCanvasDefW() const      {return fCanvasDe
fW;}
   Int_t            GetCanvasDefX() const      {return fCanvasDe
fX;}
   Int_t            GetCanvasDefY() const      {return fCanvasDe
fY;}
   Int_t            GetColorPalette(Int_t i) const;
   Int_t            GetColorModelPS() const    {return fColorMod
elPS;}
   Float_t          GetDateX()  const          {return fDateX;}
   Float_t          GetDateY() const           {return fDateY;}
   const char      *GetFitFormat()       const {return fFitForma
t.Data();}
   Int_t            GetHatchesLineWidth() const {return fHatches
LineWidth;}
   Double_t         GetHatchesSpacing() const  {return fHatchesS
pacing;}
   Width_t          GetLegendBorderSize() const   {return fLegen
dBorderSize;}
   Color_t          GetLegendFillColor() const {return fLegendFi
llColor;}
   Style_t          GetLegendFont() const {return fLegendFont;}
   Double_t         GetLegendTextSize() const {return fLegendTex
tSize;}
   Int_t            GetNumberOfColors() const;
   Color_t          GetPadColor() const        {return fPadColor
;}
   Width_t          GetPadBorderSize() const   {return fPadBorde
rSize;}
   Int_t            GetPadBorderMode() const   {return fPadBorde
rMode;}
   Float_t          GetPadBottomMargin() const {return fPadBotto
```

```
mMargin;}
   Float_t          GetPadTopMargin() const    {return fPadTopMa
rgin;}
   Float_t          GetPadLeftMargin() const   {return fPadLeftM
argin;}
   Float_t          GetPadRightMargin() const  {return fPadRight
Margin;}
   Bool_t           GetPadGridX() const        {return fPadGridX
;}
   Bool_t           GetPadGridY() const        {return fPadGridY
;}
   Int_t            GetPadTickX() const        {return fPadTickX
;}
   Int_t            GetPadTickY() const        {return fPadTickY
;}
   Color_t          GetFuncColor() const       {return fFuncColo
r;}
   Style_t          GetFuncStyle() const       {return fFuncStyl
e;}
   Width_t          GetFuncWidth() const       {return fFuncWidt
h;}
   Color_t          GetGridColor() const       {return fGridColo
r;}
   Style_t          GetGridStyle() const       {return fGridStyl
e;}
   Width_t          GetGridWidth() const       {return fGridWidt
h;}
   Color_t          GetFrameFillColor()  const {return fFrameFil
lColor;}
   Color_t          GetFrameLineColor()  const {return fFrameLin
eColor;}
   Style_t          GetFrameFillStyle()  const {return fFrameFil
lStyle;}
   Style_t          GetFrameLineStyle()  const {return fFrameLin
eStyle;}
   Width_t          GetFrameLineWidth()  const {return fFrameLin
eWidth;}
   Width_t          GetFrameBorderSize() const {return fFrameBor
derSize;}
   Int_t            GetFrameBorderMode() const {return fFrameBor
```

```
derMode;}
   Color_t          GetHistFillColor()   const {return fHistFill
Color;}
   Color_t          GetHistLineColor()   const {return fHistLine
Color;}
   Style_t          GetHistFillStyle()   const {return fHistFill
Style;}
   Style_t          GetHistLineStyle()   const {return fHistLine
Style;}
   Width_t          GetHistLineWidth()   const {return fHistLine
Width;}
   Bool_t           GetHistMinimumZero() const {return fHistMini
mumZero;}
   Double_t         GetHistTopMargin()   const {return fHistTopM
argin;}
   Float_t          GetLegoInnerR() const {return fLegoInnerR;}
   Int_t            GetNumberContours() const {return fNumberCon
tours;}
   Int_t            GetOptDate() const {return fOptDate;}
   Int_t            GetOptFile() const {return fOptFile;}
   Int_t            GetOptFit() const {return fOptFit;}
   Int_t            GetOptStat() const {return fOptStat;}
   Int_t            GetOptTitle() const {return fOptTitle;}
   Int_t            GetOptLogx() const {return fOptLogx;}
   Int_t            GetOptLogy() const {return fOptLogy;}
   Int_t            GetOptLogz() const {return fOptLogz;}
   const char      *GetPaintTextFormat() const {return fPaintTex
tFormat.Data();}
   void             GetPaperSize(Float_t &xsize, Float_t &ysize)
const;
   Int_t            GetShowEventStatus() const {return fShowEven
tStatus;}
   Int_t            GetShowEditor() const {return fShowEditor;}
   Int_t            GetShowToolBar() const {return fShowToolBar;
}

   Float_t          GetScreenFactor() const {return fScreenFacto
r;}
   Color_t          GetStatColor() const {return fStatColor;}
   Color_t          GetStatTextColor() const {return fStatTextCo
```

```
lor;}
   Width_t          GetStatBorderSize() const {return fStatBorde
rSize;}
   Style_t          GetStatFont() const  {return fStatFont;}
   Float_t          GetStatFontSize() const  {return fStatFontSi
ze;}
   Style_t          GetStatStyle() const  {return fStatStyle;}
   const char      *GetStatFormat() const {return fStatFormat.Da
ta();}
   Float_t          GetStatX() const     {return fStatX;}
   Float_t          GetStatY() const     {return fStatY;}
   Float_t          GetStatW() const     {return fStatW;}
   Float_t          GetStatH() const     {return fStatH;}
   Int_t            GetStripDecimals() const {return fStripDecim
als;}
   Double_t         GetTimeOffset() const {return fTimeOffset;}
//return axis time offset
   Int_t            GetTitleAlign() {return fTitleAlign;} // ret
urn the histogram title TPaveLabel alignment
   Color_t          GetTitleFillColor() const {return fTitleColo
r;}  //return histogram title fill area color
   Color_t          GetTitleTextColor() const {return fTitleText
Color;}  //return histogram title text color
   Style_t          GetTitleStyle() const  {return fTitleStyle;}
   Float_t          GetTitleFontSize() const  {return fTitleFont
Size;} //return histogram title font size
   Width_t          GetTitleBorderSize() const {return fTitleBor
derSize;} //return border size of histogram title TPaveLabel
   Float_t          GetTitleXOffset() const {return GetTitleOffs
et("X");} //return X axis title offset
   Float_t          GetTitleXSize() const   {return GetTitleSize(
"X");}   //return X axis title size
   Float_t          GetTitleYOffset() const {return GetTitleOffs
et("Y");} //return Y axis title offset
   Float_t          GetTitleYSize() const   {return GetTitleSize(
"Y");}   //return Y axis title size
   Float_t          GetTitleX() const     {return fTitleX;}  //r
eturn left X position of histogram title TPavelabel
   Float_t          GetTitleY() const     {return fTitleY;}  //r
eturn left bottom position of histogram title TPavelabel
```

```
   Float_t          GetTitleW() const     {return fTitleW;}  //r
eturn width of histogram title TPaveLabel
   Float_t          GetTitleH() const     {return fTitleH;}  //r
eturn height of histogram title TPavelabel
   const char      *GetHeaderPS() const {return fHeaderPS.Data()
;}
   const char      *GetTitlePS()  const {return fTitlePS.Data();
}
   const char      *GetLineStyleString(Int_t i=1) const;
   Float_t          GetLineScalePS() const {return fLineScalePS;
}
   Bool_t           IsReading() const {return fIsReading;}
   virtual void     Paint(Option_t *option="");
   virtual void     Reset(Option_t *option="");

   void             SetColorModelPS(Int_t c=0);
   void             SetFitFormat(const char *format="5.4g") {fFi
tFormat = format;}
   void             SetHeaderPS(const char *header);
   void             SetHatchesLineWidth(Int_t l) {fHatchesLineWi
dth = l;}
   void             SetHatchesSpacing(Double_t h) {fHatchesSpaci
ng = TMath::Max(0.1,h);}
   void             SetTitlePS(const char *pstitle);
   void             SetLineScalePS(Float_t scale=3) {fLineScaleP
S=scale;}
   void             SetLineStyleString(Int_t i, const char *text)
;
   void             SetNdivisions(Int_t n=510, Option_t *axis="X"
);
   void             SetAxisColor(Color_t color=1, Option_t *axis=
"X");
   void             SetLabelColor(Color_t color=1, Option_t *axi
s="X");
   void             SetLabelFont(Style_t font=62, Option_t *axis=
"X");
   void             SetLabelOffset(Float_t offset=0.005, Option_
t *axis="X");
   void             SetLabelSize(Float_t size=0.04, Option_t *ax
is="X");
```

```
   void              SetLegoInnerR(Float_t rad=0.5) {fLegoInnerR
 = rad;}
   void              SetScreenFactor(Float_t factor=1) {fScreenFa
ctor = factor;}
   void              SetTickLength(Float_t length=0.03, Option_t
*axis="X");
   void              SetTitleColor(Color_t color=1, Option_t *axi
s="X"); //set axis title color or pad title color
   void              SetTitleFont(Style_t font=62, Option_t *axis=
"X"); //set axis title font or pad title font
   void              SetTitleOffset(Float_t offset=1, Option_t *a
xis="X"); //set axis title offset
   void              SetTitleSize(Float_t size=0.02, Option_t *ax
is="X");  //set axis title size or pad title size
   void              SetNumberContours(Int_t number=20);
   void              SetOptDate(Int_t datefl=1);
   void              SetOptFile(Int_t file=1) {fOptFile = file;}
   void              SetOptFit(Int_t fit=1);
   void              SetOptLogx(Int_t logx=1) {fOptLogx = logx;}
   void              SetOptLogy(Int_t logy=1) {fOptLogy = logy;}
   void              SetOptLogz(Int_t logz=1) {fOptLogz = logz;}
   void              SetOptStat(Int_t stat=1);
   void              SetOptStat(Option_t *stat);
   void              SetOptTitle(Int_t tit=1) {fOptTitle = tit;}
   void              SetBarOffset(Float_t baroff=0.5) {fBarOffset
 = baroff;}
   void              SetBarWidth(Float_t barwidth=0.5) {fBarWidth
 = barwidth;}
   void              SetDateX(Float_t x=0.01) {fDateX = x;}
   void              SetDateY(Float_t y=0.01) {fDateY = y;}
   void              SetEndErrorSize(Float_t np=2);
   void              SetErrorX(Float_t errorx=0.5) {fErrorX = err
orx;}
   void              SetCanvasPreferGL(Bool_t prefer = kTRUE) {fC
anvasPreferGL=prefer;}
   void              SetDrawBorder(Int_t drawborder=1) {fDrawBord
er = drawborder;}
   void              SetCanvasColor(Color_t color=19) {fCanvasCol
or = color;}
   void              SetCanvasBorderSize(Width_t size=1) {fCanvas
```

```
BorderSize = size;}
   void              SetCanvasBorderMode(Int_t mode=1) {fCanvasBo
rderMode = mode;}
   void              SetCanvasDefH(Int_t h=500) {fCanvasDefH = h;
}
   void              SetCanvasDefW(Int_t w=700) {fCanvasDefW = w;
}
   void              SetCanvasDefX(Int_t topx=10) {fCanvasDefX =
topx;}
   void              SetCanvasDefY(Int_t topy=10) {fCanvasDefY =
topy;}
   void              SetLegendBorderSize(Width_t size=4) {fLegend
BorderSize = size;}
   void              SetLegendFillColor(Color_t color=0) {fLegend
FillColor = color;}
   void              SetLegendFont(Style_t font=62) {fLegendFont
= font;}
   void              SetLegendTextSize(Double_t size=0.) {fLegend
TextSize = size;}
   void              SetPadColor(Color_t color=19) {fPadColor = c
olor;}
   void              SetPadBorderSize(Width_t size=1) {fPadBorder
Size = size;}
   void              SetPadBorderMode(Int_t mode=1) {fPadBorderMo
de = mode;}
   void              SetPadBottomMargin(Float_t margin=0.1) {fPad
BottomMargin=margin;}
   void              SetPadTopMargin(Float_t margin=0.1)    {fPad
TopMargin=margin;}
   void              SetPadLeftMargin(Float_t margin=0.1)   {fPad
LeftMargin=margin;}
   void              SetPadRightMargin(Float_t margin=0.1)  {fPad
RightMargin=margin;}
   void              SetPadGridX(Bool_t gridx) {fPadGridX = gridx
;}
   void              SetPadGridY(Bool_t gridy) {fPadGridY = gridy
;}
   void              SetPadTickX(Int_t tickx)  {fPadTickX = tickx
;}
   void              SetPadTickY(Int_t ticky)  {fPadTickY = ticky
```

```
;}
   void              SetFuncStyle(Style_t style=1) {fFuncStyle =
style;}
   void              SetFuncColor(Color_t color=1) {fFuncColor =
color;}
   void              SetFuncWidth(Width_t width=4) {fFuncWidth =
width;}
   void              SetGridStyle(Style_t style=3) {fGridStyle =
style;}
   void              SetGridColor(Color_t color=0) {fGridColor =
color;}
   void              SetGridWidth(Width_t width=1) {fGridWidth =
width;}
   void              SetFrameFillColor(Color_t color=1) {fFrameFi
llColor = color;}
   void              SetFrameLineColor(Color_t color=1) {fFrameLi
neColor = color;}
   void              SetFrameFillStyle(Style_t styl=0)  {fFrameFi
llStyle = styl;}
   void              SetFrameLineStyle(Style_t styl=0)  {fFrameLi
neStyle = styl;}
   void              SetFrameLineWidth(Width_t width=1) {fFrameLi
neWidth = width;}
   void              SetFrameBorderSize(Width_t size=1) {fFrameBo
rderSize = size;}
   void              SetFrameBorderMode(Int_t mode=1) {fFrameBord
erMode = mode;}
   void              SetHistFillColor(Color_t color=1) {fHistFill
Color = color;}
   void              SetHistLineColor(Color_t color=1) {fHistLine
Color = color;}
   void              SetHistFillStyle(Style_t styl=0)  {fHistFill
Style = styl;}
   void              SetHistLineStyle(Style_t styl=0)  {fHistLine
Style = styl;}
   void              SetHistLineWidth(Width_t width=1) {fHistLine
Width = width;}
   void              SetHistMinimumZero(Bool_t zero=kTRUE);
   void              SetHistTopMargin(Double_t hmax=0.05) {fHistT
opMargin = hmax;}
```

```
   void              SetPaintTextFormat(const char *format="g") {
fPaintTextFormat = format;}
   void              SetPaperSize(EPaperSize size);
   void              SetPaperSize(Float_t xsize=20, Float_t ysize=
26);
   void              SetStatColor(Color_t color=19) {fStatColor=c
olor;}
   void              SetStatTextColor(Color_t color=1) {fStatText
Color=color;}
   void              SetStatStyle(Style_t style=1001) {fStatStyle
=style;}
   void              SetStatBorderSize(Width_t size=2) {fStatBord
erSize=size;}
   void              SetStatFont(Style_t font=62) {fStatFont=font
;}
   void              SetStatFontSize(Float_t size=0)  {fStatFontS
ize=size;}
   void              SetStatFormat(const char *format="6.4g") {fS
tatFormat = format;}
   void              SetStatX(Float_t x=0)    {fStatX=x;}
   void              SetStatY(Float_t y=0)    {fStatY=y;}
   void              SetStatW(Float_t w=0.19) {fStatW=w;}
   void              SetStatH(Float_t h=0.1)  {fStatH=h;}
   void              SetStripDecimals(Bool_t strip=kTRUE);
   void              SetTimeOffset(Double_t toffset);
   void              SetTitleAlign(Int_t a=13) {fTitleAlign=a;}
   void              SetTitleFillColor(Color_t color=1)   {fTitle
Color=color;}
   void              SetTitleTextColor(Color_t color=1)   {fTitle
TextColor=color;}
   void              SetTitleStyle(Style_t style=1001)  {fTitleSt
yle=style;}
   void              SetTitleFontSize(Float_t size=0)   {fTitleFo
ntSize=size;}
   void              SetTitleBorderSize(Width_t size=2) {fTitleBo
rderSize=size;}
   void              SetTitleXOffset(Float_t offset=1)  {SetTitle
Offset(offset,"X");}
   void              SetTitleXSize(Float_t size=0.02)   {SetTitle
Size(size,"X");}
```

```
   void              SetTitleYOffset(Float_t offset=1)  {SetTitle
Offset(offset,"Y");}
   void              SetTitleYSize(Float_t size=0.02)   {SetTitle
Size(size,"Y");}
   void              SetTitleX(Float_t x=0)     {fTitleX=x;}
   void              SetTitleY(Float_t y=0.985) {fTitleY=y;}
   void              SetTitleW(Float_t w=0)     {fTitleW=w;}
   void              SetTitleH(Float_t h=0)     {fTitleH=h;}
   void              ToggleEventStatus() { fShowEventStatus = fSh
owEventStatus ? 0 : 1; }
   void              ToggleEditor() { fShowEditor = fShowEditor ?
0 : 1; }
   void              ToggleToolBar() { fShowToolBar = fShowToolBa
r ? 0 : 1; }
   void              SetIsReading(Bool_t reading=kTRUE);
   void              SetPalette(Int_t ncolors=kBird, Int_t *color
s=0, Float_t alpha=1.);
   void              SavePrimitive(std::ostream &out, Option_t *
= "");
   void              SaveSource(const char *filename, Option_t *o
ption=0);
```

TSystem

class

继承TNamed

```
struct FileStat_t {
   Long_t   fDev;          // device id
   Long_t   fIno;          // inode
   Int_t    fMode;         // protection (combination of EFileMo
deMask bits)
   Int_t    fUid;          // user id of owner
   Int_t    fGid;          // group id of owner
   Long64_t fSize;         // total size in bytes
   Long_t   fMtime;        // modification date
   Bool_t   fIsLink;       // symbolic link
   TString  fUrl;          // end point url of file
   FileStat_t() : fDev(0), fIno(0), fMode(0), fUid(0), fGid(0),
fSize(0),
                  fMtime(0), fIsLink(kFALSE), fUrl("") { }
};

struct UserGroup_t {
   Int_t    fUid;          // user id
   Int_t    fGid;          // group id
   TString  fUser;         // user name
   TString  fGroup;        // group name
   TString  fPasswd;       // password
   TString  fRealName;     // user full name
   TString  fShell;        // user preferred shell
   UserGroup_t() : fUid(0), fGid(0), fUser(), fGroup(), fPasswd(
),
                   fRealName (), fShell() { }
};

struct SysInfo_t {
   TString   fOS;          // OS
```

```
   TString   fModel;       // computer model
   TString   fCpuType;     // type of cpu
   Int_t     fCpus;        // number of cpus
   Int_t     fCpuSpeed;    // cpu speed in MHz
   Int_t     fBusSpeed;    // bus speed in MHz
   Int_t     fL2Cache;     // level 2 cache size in KB
   Int_t     fPhysRam;     // physical RAM in MB
   SysInfo_t() : fOS(), fModel(), fCpuType(), fCpus(0), fCpuSpee
d(0),
                 fBusSpeed(0), fL2Cache(0), fPhysRam(0) { }
   virtual ~SysInfo_t() { }
   ClassDef(SysInfo_t, 1); // System information - OS, CPU, RAM.
};

struct CpuInfo_t {
   Float_t   fLoad1m;      // cpu load average over 1 m
   Float_t   fLoad5m;      // cpu load average over 5 m
   Float_t   fLoad15m;     // cpu load average over 15 m
   Float_t   fUser;        // cpu user load in percentage
   Float_t   fSys;         // cpu sys load in percentage
   Float_t   fTotal;       // cpu user+sys load in percentage
   Float_t   fIdle;        // cpu idle percentage
   CpuInfo_t() : fLoad1m(0), fLoad5m(0), fLoad15m(0),
                 fUser(0), fSys(0), fTotal(0), fIdle(0) { }
   virtual ~CpuInfo_t() { }
   ClassDef(CpuInfo_t, 1); // CPU load information.
};

struct MemInfo_t {
   Int_t     fMemTotal;    // total RAM in MB
   Int_t     fMemUsed;     // used RAM in MB
   Int_t     fMemFree;     // free RAM in MB
   Int_t     fSwapTotal;   // total swap in MB
   Int_t     fSwapUsed;    // used swap in MB
   Int_t     fSwapFree;    // free swap in MB
   MemInfo_t() : fMemTotal(0), fMemUsed(0), fMemFree(0),
                 fSwapTotal(0), fSwapUsed(0), fSwapFree(0) { }
   virtual ~MemInfo_t() { }
   ClassDef(MemInfo_t, 1); // Memory utilization information.
};
```

```
struct ProcInfo_t {
   Float_t   fCpuUser;     // user time used by this process in
seconds
   Float_t   fCpuSys;      // system time used by this process i
n seconds
   Long_t    fMemResident; // resident memory used by this proce
ss in KB
   Long_t    fMemVirtual;  // virtual memory used by this proces
s in KB
   ProcInfo_t() : fCpuUser(0), fCpuSys(0), fMemResident(0),
                  fMemVirtual(0) { }
   virtual ~ProcInfo_t() { }
   ClassDef(ProcInfo_t, 1);// System resource usage of given pro
cess.
};

struct RedirectHandle_t {
   TString   fFile;        // File where the output was redirect
ed
   TString   fStdOutTty;   // tty associated with stdout, if any
 (e.g. from ttyname(...))
   TString   fStdErrTty;   // tty associated with stderr, if any
 (e.g. from ttyname(...))
   Int_t     fStdOutDup;   // Duplicated descriptor for stdout
   Int_t     fStdErrDup;   // Duplicated descriptor for stderr
   Int_t     fReadOffSet;  // Offset where to start reading the
file (used by ShowOutput(...))
   RedirectHandle_t(const char *n = 0) : fFile(n), fStdOutTty(),
 fStdErrTty(), fStdOutDup(-1),
                                          fStdErrDup(-1), fReadOf
fSet(-1) { }
   void Reset() { fFile = ""; fStdOutTty = ""; fStdErrTty = "";
                  fStdOutDup = -1; fStdErrDup = -1; fReadOffSet
= -1; }
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........o
ooOOOOOooo......
```

```
TSystem(const char *name = "Generic", const char *title = "Gener
ic System");
virtual ~TSystem();

//---- Misc
virtual Bool_t          Init();
virtual void            SetProgname(const char *name);
virtual void            SetDisplay();
void                    SetErrorStr(const char *errstr);
const char             *GetErrorStr() const { return GetLastErro
rString(); }
virtual const char     *GetError();
void                    RemoveOnExit(TObject *obj);
virtual const char     *HostName();
virtual void            NotifyApplicationCreated();

static Int_t            GetErrno();
static void             ResetErrno();
void                    Beep(Int_t freq=-1, Int_t duration=-1, B
ool_t setDefault=kFALSE);
void                    GetBeepDefaults(Int_t &freq, Int_t &dura
tion) const { freq = fBeepFreq; duration = fBeepDuration; }

//---- EventLoop
virtual void            Run();
virtual Bool_t          ProcessEvents();
virtual void            DispatchOneEvent(Bool_t pendingOnly = kF
ALSE);
virtual void            ExitLoop();
Bool_t                  InControl() const { return fInControl; }
virtual void            InnerLoop();
virtual Int_t           Select(TList *active, Long_t timeout);
virtual Int_t           Select(TFileHandler *fh, Long_t timeout)
;

//---- Handling of system events
virtual void            AddSignalHandler(TSignalHandler *sh);
virtual TSignalHandler *RemoveSignalHandler(TSignalHandler *sh);
virtual void            ResetSignal(ESignals sig, Bool_t reset =
 kTRUE);
```

```
virtual void            ResetSignals();
virtual void            IgnoreSignal(ESignals sig, Bool_t ignore
 = kTRUE);
virtual void            IgnoreInterrupt(Bool_t ignore = kTRUE);
virtual TSeqCollection *GetListOfSignalHandlers() const { return
 fSignalHandler; }
virtual void            AddFileHandler(TFileHandler *fh);
virtual TFileHandler   *RemoveFileHandler(TFileHandler *fh);
virtual TSeqCollection *GetListOfFileHandlers() const { return f
FileHandler; }
virtual void            AddStdExceptionHandler(TStdExceptionHand
ler *eh);
virtual TStdExceptionHandler *RemoveStdExceptionHandler(TStdExce
ptionHandler *eh);
virtual TSeqCollection *GetListOfStdExceptionHandlers() const {
return fStdExceptionHandler; }

//---- Floating Point Exceptions Control
virtual Int_t           GetFPEMask();
virtual Int_t           SetFPEMask(Int_t mask = kDefaultMask);

//---- Time & Date
virtual TTime           Now();
virtual TSeqCollection *GetListOfTimers() const { return fTimers
; }
virtual void            AddTimer(TTimer *t);
virtual TTimer         *RemoveTimer(TTimer *t);
virtual void            ResetTimer(TTimer *) { }
virtual Long_t          NextTimeOut(Bool_t mode);
virtual void            Sleep(UInt_t milliSec);//程序休眠

//---- Processes
virtual Int_t           Exec(const char *shellcmd);//执行命令
virtual FILE           *OpenPipe(const char *command, const char
 *mode);
virtual int             ClosePipe(FILE *pipe);
virtual TString         GetFromPipe(const char *command);
virtual void            Exit(int code, Bool_t mode = kTRUE);//退
出程序
virtual void            Abort(int code = 0);
```

```
virtual int             GetPid();
virtual void            StackTrace();

//---- Directories
virtual int             MakeDirectory(const char *name);//新建文
件夹
virtual void           *OpenDirectory(const char *name);//打开文
件夹
virtual void            FreeDirectory(void *dirp);
virtual const char     *GetDirEntry(void *dirp);
virtual void           *GetDirPtr() const { return 0; }
virtual Bool_t          ChangeDirectory(const char *path);//进入
该目录
virtual const char     *WorkingDirectory();
virtual const char     *HomeDirectory(const char *userName = 0);
virtual int             mkdir(const char *name, Bool_t recursive
 = kFALSE);
Bool_t                  cd(const char *path) { return ChangeDire
ctory(path); }//进入该目录
const char             *pwd() { return WorkingDirectory(); }//获
得当前路径
virtual const char     *TempDirectory() const;
virtual FILE           *TempFileName(TString &base, const char *
dir = 0);

//---- Paths & Files
virtual const char     *BaseName(const char *pathname);
virtual const char     *DirName(const char *pathname);
virtual char           *ConcatFileName(const char *dir, const ch
ar *name);
virtual Bool_t          IsAbsoluteFileName(const char *dir);
virtual Bool_t          IsFileInIncludePath(const char *name, ch
ar **fullpath = 0);
virtual const char     *PrependPathName(const char *dir, TString
& name);
virtual Bool_t          ExpandPathName(TString &path);
virtual char           *ExpandPathName(const char *path);
virtual Bool_t          AccessPathName(const char *path, EAccess
Mode mode = kFileExists);//判断该文件是否存在、是否可写、是否可读
virtual Bool_t          IsPathLocal(const char *path);
```

```
virtual int             CopyFile(const char *from, const char *t
o, Bool_t overwrite = kFALSE);//复制文件
virtual int             Rename(const char *from, const char *to);
//文件、文件夹重命名
virtual int             Link(const char *from, const char *to);
virtual int             Symlink(const char *from, const char *to)
;
virtual int             Unlink(const char *name);
int                     GetPathInfo(const char *path, Long_t *id
, Long_t *size, Long_t *flags, Long_t *modtime);
int                     GetPathInfo(const char *path, Long_t *id
, Long64_t *size, Long_t *flags, Long_t *modtime);
virtual int             GetPathInfo(const char *path, FileStat_t
 &buf);//获取该文件信息
virtual int             GetFsInfo(const char *path, Long_t *id,
Long_t *bsize, Long_t *blocks, Long_t *bfree);
virtual int             Chmod(const char *file, UInt_t mode);
virtual int             Umask(Int_t mask);
virtual int             Utime(const char *file, Long_t modtime,
Long_t actime);
virtual const char     *UnixPathName(const char *unixpathname);/
/当前文件所在目录
virtual const char     *FindFile(const char *search, TString& fi
le, EAccessMode mode = kFileExists);
virtual char           *Which(const char *search, const char *fi
le, EAccessMode mode = kFileExists);//寻找可执行文件所在位置
virtual TList          *GetVolumes(Option_t *) const { return 0;
 }

//---- Users & Groups
virtual Int_t           GetUid(const char *user = 0);
virtual Int_t           GetGid(const char *group = 0);
virtual Int_t           GetEffectiveUid();
virtual Int_t           GetEffectiveGid();
virtual UserGroup_t    *GetUserInfo(Int_t uid);
virtual UserGroup_t    *GetUserInfo(const char *user = 0);
virtual UserGroup_t    *GetGroupInfo(Int_t gid);
virtual UserGroup_t    *GetGroupInfo(const char *group = 0);

//---- Environment Manipulation
```

```
virtual void            Setenv(const char *name, const char *val
ue);
virtual void            Unsetenv(const char *name);
virtual const char     *Getenv(const char *env);//获取环境变量对应
的路径

//---- System Logging
virtual void            Openlog(const char *name, Int_t options,
 ELogFacility facility);
virtual void            Syslog(ELogLevel level, const char *mess)
;
virtual void            Closelog();

//---- Standard Output redirection
virtual Int_t           RedirectOutput(const char *name, const c
har *mode = "a", RedirectHandle_t *h = 0);
virtual void            ShowOutput(RedirectHandle_t *h);

//---- Dynamic Loading
virtual void            AddDynamicPath(const char *pathname);
virtual const char     *GetDynamicPath();
virtual void            SetDynamicPath(const char *pathname);
char                   *DynamicPathName(const char *lib, Bool_t
quiet = kFALSE);
virtual const char     *FindDynamicLibrary(TString& lib, Bool_t
quiet = kFALSE);
virtual Func_t          DynFindSymbol(const char *module, const
char *entry);
virtual int             Load(const char *module, const char *ent
ry = "", Bool_t system = kFALSE);//加载文件、链接库等
virtual void            Unload(const char *module);
virtual void            ListSymbols(const char *module, const ch
ar *re = "");
virtual void            ListLibraries(const char *regexp = "");
virtual const char     *GetLibraries(const char *regexp = "",
                      const char *option = "",
                      Bool_t isRegexp = kTRUE);

//---- RPC
virtual TInetAddress    GetHostByName(const char *server);
```

```
virtual TInetAddress    GetPeerName(int sock);
virtual TInetAddress    GetSockName(int sock);
virtual int             GetServiceByName(const char *service);
virtual char           *GetServiceByPort(int port);
virtual int             OpenConnection(const char *server, int p
ort, int tcpwindowsize = -1, const char *protocol = "tcp");
virtual int             AnnounceTcpService(int port, Bool_t reus
e, int backlog, int tcpwindowsize = -1);
virtual int             AnnounceUdpService(int port, int backlog)
;
virtual int             AnnounceUnixService(int port, int backlo
g);
virtual int             AnnounceUnixService(const char *sockpath
, int backlog);
virtual int             AcceptConnection(int sock);
virtual void            CloseConnection(int sock, Bool_t force =
 kFALSE);
virtual int             RecvRaw(int sock, void *buffer, int leng
th, int flag);
virtual int             SendRaw(int sock, const void *buffer, int
 length, int flag);
virtual int             RecvBuf(int sock, void *buffer, int leng
th);
virtual int             SendBuf(int sock, const void *buffer, int
 length);
virtual int             SetSockOpt(int sock, int kind, int val);
virtual int             GetSockOpt(int sock, int kind, int *val)
;

//---- System, CPU and Memory info
virtual int             GetSysInfo(SysInfo_t *info) const;
virtual int             GetCpuInfo(CpuInfo_t *info, Int_t sample
Time = 1000) const;//获取CPU信息
virtual int             GetMemInfo(MemInfo_t *info) const;//获取
内存使用情况信息
virtual int             GetProcInfo(ProcInfo_t *info) const;

//---- ACLiC (Automatic Compiler of Shared Library for CINT)
virtual void            AddIncludePath(const char *includePath);
virtual void            AddLinkedLibs(const char *linkedLib);
```

```
virtual int             CompileMacro(const char *filename, Optio
n_t *opt="", const char* library_name = "", const char* build_di
r = "", UInt_t dirmode = 0);//执行宏文件？？？
virtual Int_t           GetAclicProperties() const;
virtual const char     *GetBuildArch() const;
virtual const char     *GetBuildCompiler() const;
virtual const char     *GetBuildCompilerVersion() const;
virtual const char     *GetBuildNode() const;
virtual const char     *GetBuildDir() const;
virtual const char     *GetFlagsDebug() const;
virtual const char     *GetFlagsOpt() const;
virtual const char     *GetIncludePath();
virtual const char     *GetLinkedLibs() const;
virtual const char     *GetLinkdefSuffix() const;
virtual EAclicMode      GetAclicMode() const;
virtual const char     *GetMakeExe() const;
virtual const char     *GetMakeSharedLib() const;
virtual const char     *GetSoExt() const;
virtual const char     *GetObjExt() const;
virtual void            SetBuildDir(const char* build_dir, Bool_
t isflat = kFALSE);
virtual void            SetFlagsDebug(const char *);
virtual void            SetFlagsOpt(const char *);
virtual void            SetIncludePath(const char *includePath);
virtual void            SetMakeExe(const char *directives);
virtual void            SetAclicMode(EAclicMode mode);
virtual void            SetMakeSharedLib(const char *directives)
;
virtual void            SetLinkedLibs(const char *linkedLibs);
virtual void            SetLinkdefSuffix(const char *suffix);
virtual void            SetSoExt(const char *soExt);
virtual void            SetObjExt(const char *objExt);
virtual TString         SplitAclicMode(const char *filename, TSt
ring &mode, TString &args, TString &io) const;
virtual void            CleanCompiledMacros();
```

code

TText

继承 TNamed, TAttText, TAttBBox2D

Base class for several text objects.

See TAttText for a list of text attributes or fonts, and also for a discussion on text speed and font quality.

By default, the text is drawn in the pad coordinates system. One can draw in NDC coordinates [0,1] if the function SetNDC is called for a TText object.

class

```
   TText();
   TText(Double_t x, Double_t y, const char *text);
   TText(Double_t x, Double_t y, const wchar_t *text);
   TText(const TText &text);
   virtual ~TText();
   void             Copy(TObject &text) const;
/// Copy this text to text.

   virtual Int_t    DistancetoPrimitive(Int_t px, Int_t py);
/// Compute distance from point px,py to a string.
/// The rectangle surrounding this string is evaluated.
/// If the point (px,py) is in the rectangle, the distance is se
t to zero.

   virtual TText   *DrawText(Double_t x, Double_t y, const char
*text);
/// Draw this text with new coordinates.

   virtual TText   *DrawText(Double_t x, Double_t y, const wchar
_t *text);
/// Draw this text with new coordinates.

   virtual TText   *DrawTextNDC(Double_t x, Double_t y, const ch
```

```
ar *text);
/// Draw this text with new coordinates in NDC.

   virtual TText   *DrawTextNDC(Double_t x, Double_t y, const wc
har_t *text);
/// Draw this text with new coordinates in NDC.

   virtual void     ExecuteEvent(Int_t event, Int_t px, Int_t py)
;
/// Execute action corresponding to one event.
///  This member function must be implemented to realize the act
ion
///  corresponding to the mouse click on the object in the window


   virtual void     GetControlBox(Int_t x, Int_t y, Double_t the
ta,
                                  Int_t cBoxX[4], Int_t cBoxY[4])
;
/// Return the text control box. The text position coordinates i
s (x,y) and
/// the text angle is theta. The control box coordinates are ret
urned in cBoxX
/// and cBoxY.

   Double_t         GetX() const  { return fX; }
   virtual void     GetBoundingBox(UInt_t &w, UInt_t &h, Bool_t
angle = kFALSE);
/// Return text size in pixels. By default the size returned doe
s not take
/// into account the text angle (angle = kFALSE). If angle is se
t to kTRUE
/// w and h take the angle into account.

   virtual void     GetTextAscentDescent(UInt_t &a, UInt_t &d, c
onst char *text) const;
/// Return text ascent and descent for string text
///  - in a return total text ascent
///  - in d return text descent
```

```
   virtual void     GetTextAscentDescent(UInt_t &a, UInt_t &d, c
onst wchar_t *text) const;
/// Return text ascent and descent for string text
///  - in a return total text ascent
///  - in d return text descent

   virtual void     GetTextExtent(UInt_t &w, UInt_t &h, const ch
ar *text) const;
/// Return text extent for string text
///  - in w return total text width
///  - in h return text height

   virtual void     GetTextExtent(UInt_t &w, UInt_t &h, const wc
har_t *text) const;
/// Return text extent for string text
///  - in w return total text width
///  - in h return text height

   virtual void     GetTextAdvance(UInt_t &a, const char *text,
const Bool_t kern=kTRUE) const;
/// Return text advance for string text
/// if kern is true (default) kerning is taken into account. If
it is false
/// the kerning is not taken into account.

   const void *     GetWcsTitle(void) const;
/// Returns the text as UNICODE.

   Double_t         GetY() const  { return fY; }

   virtual void     ls(Option_t *option="") const;
/// List this text with its attributes.

   virtual void     Paint(Option_t *option="");
/// Paint this text with its current attributes.

   virtual void     PaintControlBox(Int_t x, Int_t y, Double_t t
heta);
/// Paint the text control box. (x,y) are the coordinates where
the control
```

```
/// box should be painted and theta is the angle of the box.

   virtual void     PaintText(Double_t x, Double_t y, const char
 *text);
/// Draw this text with new coordinates.

   virtual void     PaintText(Double_t x, Double_t y, const wcha
r_t *text);
/// Draw this text with new coordinates.

   virtual void     PaintTextNDC(Double_t u, Double_t v, const c
har *text);
/// Draw this text with new coordinates in NDC.

   virtual void     PaintTextNDC(Double_t u, Double_t v, const w
char_t *text);
/// Draw this text with new coordinates in NDC.

   virtual void     Print(Option_t *option="") const;
/// Dump this text with its attributes.

   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
/// Save primitive as a C++ statement(s) on output stream out

   virtual void     SetMbTitle(const wchar_t *title=L""); // *ME
NU*
/// Change (i.e. set) the title of the TNamed.

   virtual void     SetNDC(Bool_t isNDC=kTRUE);
/// Set NDC mode on if isNDC = kTRUE, off otherwise

   virtual void     SetText(Double_t x, Double_t y, const char *
text) {fX=x; fY=y; SetTitle(text);} // *MENU* *ARGS={x=>fX,y=>fY
,text=>fTitle}
   virtual void     SetText(Double_t x, Double_t y, const wchar_
t *text) {fX=x; fY=y; SetMbTitle(text);}
   virtual void     SetX(Double_t x) { fX = x; } // *MENU*
   virtual void     SetY(Double_t y) { fY = y; } // *MENU*
```

```
   virtual Rectangle_t  GetBBox();
/// Return the "bounding Box" of the Box

   virtual TPoint       GetBBoxCenter();
/// Return the point given by Alignment as 'center'

   virtual void         SetBBoxCenter(const TPoint &p);
/// Set the point given by Alignment as 'center'

   virtual void         SetBBoxCenterX(const Int_t x);
/// Set X coordinate of the point given by Alignment as 'center'

   virtual void         SetBBoxCenterY(const Int_t y);
/// Set Y coordinate of the point given by Alignment as 'center'

   virtual void         SetBBoxX1(const Int_t x); //Not Implemen
ted
   virtual void         SetBBoxX2(const Int_t x); //Not Implemen
ted
   virtual void         SetBBoxY1(const Int_t y); //Not Implemen
ted
   virtual void         SetBBoxY2(const Int_t y); //Not Implemen
ted
```

code

```
// By default, the text is drawn in the pad coordinates system.
One can draw in NDC coordinates [0,1] if the function SetNDC is
called for a TText object.
   TText *t = new TText(.5,.5,"Hello World !");
   t->SetTextAlign(22);
   t->SetTextColor(kRed+2);
   t->SetTextFont(43);
   t->SetTextSize(40);
   t->SetTextAngle(45);
   t->Draw();
```

example

TThread

TThreadFactory

TThreadImp

TTime

基类

Basic time type with millisecond precision.

class

```
TTime(): fMilliSec(0) { }
TTime(Long64_t msec): fMilliSec(msec) { }
TTime(const TTime &t): fMilliSec(t.fMilliSec) { }
virtual ~TTime() { }

TTime& operator=(const TTime &t);

TTime operator+=(const TTime &t);
TTime operator-=(const TTime &t);
TTime operator*=(const TTime &t);
TTime operator/=(const TTime &t);

friend TTime operator+(const TTime &t1, const TTime &t2);
friend TTime operator-(const TTime &t1, const TTime &t2);
friend TTime operator*(const TTime &t1, const TTime &t2);
friend TTime operator/(const TTime &t1, const TTime &t2);

friend Bool_t operator== (const TTime &t1, const TTime &t2);
friend Bool_t operator!= (const TTime &t1, const TTime &t2);
friend Bool_t operator<  (const TTime &t1, const TTime &t2);
friend Bool_t operator<= (const TTime &t1, const TTime &t2);
friend Bool_t operator>  (const TTime &t1, const TTime &t2);
friend Bool_t operator>= (const TTime &t1, const TTime &t2);

operator long() const;
operator unsigned long() const;
operator long long() const;
operator unsigned long long() const;
const char *AsString() const;
```

code

example

TTimer

继承 TSysEvtHandler

可用于图形界面

Handles synchronous and a-synchronous timer events. You can use this class in one of the following ways:

- Sub-class TTimer and override the Notify() method.
- Re-implement the TObject::HandleTimer() method in your class and pass a pointer to this object to timer, see the SetObject() method.
- Pass an interpreter command to timer, see SetCommand() method.
- Create a TTimer, connect its Timeout() signal to the appropriate methods. Then when the time is up it will emit a Timeout() signal and call connected slots.

Minimum timeout interval is defined in TSystem::ESysConstants as kItimerResolution (currently 10 ms).

class

```
   TTimer(Long_t milliSec = 0, Bool_t mode = kTRUE);
   TTimer(TObject *obj, Long_t milliSec, Bool_t mode = kTRUE);
   TTimer(const char *command, Long_t milliSec, Bool_t mode = kT
RUE);
   virtual ~TTimer() { Remove(); }

   Bool_t          CheckTimer(const TTime &now);
   const char     *GetCommand() const { return fCommand.Data(); }
   TObject        *GetObject() { return fObject; }
   TTime           GetTime() const { return fTime; }
   UInt_t          GetTimerID() { return fTimeID;}
   TTime           GetAbsTime() const { return fAbsTime; }
   Bool_t          HasTimedOut() const { return fTimeout; }
   Bool_t          IsSync() const { return fSync; }
   Bool_t          IsAsync() const { return !fSync; }
```

```
   Bool_t          IsInterruptingSyscalls() const { return fIntSy
scalls; }
   virtual Bool_t Notify();
   void            Add() { TurnOn(); }
   void            Remove() { TurnOff(); }
   void            Reset();
   void            SetCommand(const char *command);
   void            SetObject(TObject *object);
   void            SetInterruptSyscalls(Bool_t set = kTRUE);
   void            SetTime(Long_t milliSec) { fTime = milliSec; }
   void            SetTimerID(UInt_t id = 0) { fTimeID = id; }
   virtual void    Start(Long_t milliSec = -1, Bool_t singleShot
= kFALSE);
   // Starts the timer with a milliSec timeout. If milliSec is 0
   // then the timeout will be the minimum timeout (see TSystem:
:ESysConstants,
   // i.e. 10 ms), if milliSec is -1 then the time interval as p
reviously
   // specified (in ctor or SetTime()) will be used.
   // If singleShot is kTRUE, the timer will be activated only o
nce,
   // otherwise it will continue until it is stopped.
   // See also TurnOn(), Stop(), TurnOff().

   virtual void    Stop() { TurnOff(); }
   virtual void    TurnOn();                          //*SIGNAL*
   // Add the timer to the system timer list. If a TTimer subcla
ss has to be
   // placed on another list, override TurnOn() to add the timer
 to the correct
   // list.

   virtual void    TurnOff();                         //*SIGNAL*
   // Remove timer from system timer list. This requires that a
timer
   // has been placed in the system timer list (using TurnOn()).
   // If a TTimer subclass is placed on another list, override T
urnOff() to
   // remove the timer from the correct list.
```

```
   virtual void   Timeout() { Emit("Timeout()"); }  //*SIGNAL*

   static void    SingleShot(Int_t milliSec, const char *receive
r_class,
                             void *receiver, const char *method)
;
   // This static function calls a slot after a given time inter
val.
   // Created internal timer will be deleted after that.
```

code

```
//  Signal/slots example:
TTimer *timer = new TTimer();
timer->Connect("Timeout()", "myObjectClassName", myObject, "Time
rDone()");
timer->Start(2000, kTRUE);   // 2 seconds single-shot

// Timeout signal is emitted repeadetly with minimum timeout
// timer->Start(0, kFALSE);
```

example

```
// demo of Timers
Int_t i;
Float_t ratio;
TSlider *slider;
TCanvas *c1;
void hsumTimer(Int_t nfill=100000)
{
// Simple example illustrating how to use the C++ interpreter
// to fill histograms in a loop and show the graphics results
// This program is a variant of the tutorial "hsum".
// It illustrates the use of Timers.
  c1 = new TCanvas("c1","The HSUM example",200,10,600,400);
  c1->SetGrid();
```

```
// Create some histograms.
  total  = new TH1F("total","This is the total distribution",100,
-4,4);
  main   = new TH1F("main","Main contributor",100,-4,4);
  s1     = new TH1F("s1","This is the first signal",100,-4,4);
  s2     = new TH1F("s2","This is the second signal",100,-4,4);
  total->Sumw2();   // store the sum of squares of weights
  total->SetMarkerStyle(21);
  total->SetMarkerSize(0.7);
  main->SetFillColor(16);
  s1->SetFillColor(42);
  s2->SetFillColor(46);
  total->SetMaximum(nfill/20.);
  total->Draw("e1p");
  main->Draw("same");
  s1->Draw("same");
  s2->Draw("same");
  c1->Update();slider = new TSlider("slider",
     "test",4.2,0,4.6,0.8*total->GetMaximum(),38);
  slider->SetFillColor(46);

// Create a TTimer (hsumUpdate called every 30 msec)
  TTimer timer("hsumUpdate()",30);
  timer.TurnOn();

 // Fill histograms randomly
  Float_t xs1, xs2, xmain;
  gRandom->SetSeed();
  for (Int_t i=0; i<nfill; i++) {
     ratio = Float_t(i)/Float_t(nfill);
     if (gSystem->ProcessEvents()) break;
     xmain = gRandom->Gaus(-1,1.5);
     xs1   = gRandom->Gaus(-0.5,0.5);
     xs2   = gRandom->Landau(1,0.15);
     main->Fill(xmain);
     s1->Fill(xs1,0.3);
     s2->Fill(xs2,0.2);
     total->Fill(xmain);
     total->Fill(xs1,0.3);
```

```
      total->Fill(xs2,0.2);
    }
    timer.TurnOff();
    hsumUpdate();
  }

  void hsumUpdate()
  {
  // called when Timer times out
     if (slider) slider->SetRange(0,ratio);
     c1->Modified();
     c1->Update();
  }
```

TTimeStamp

// The TTimeStamp encapsulates seconds and ns since EPOCH // // This extends (and isolates) struct timespec // struct timespec // { // time_t tv_sec; / *seconds* / // long tv_nsec; / *nanoseconds* / // } // time_t seconds is relative to Jan 1, 1970 00:00:00 UTC // // No accounting of leap seconds is made. // // Due to ROOT/CINT limitations TTimeStamp does not explicitly // hold a timespec struct; attempting to do so means the Streamer // must be hand written. Instead we have chosen to simply contain // similar fields within the private area of this class. // // NOTE: the use of time_t (and its default implementation as a 32 int) // implies overflow conditions occurs somewhere around // Jan 18, 19:14:07, 2038. // If this experiment is still going when it becomes significant // someone will have to deal with it.

class

```
   // empty ctor (builds current time with nsec field incremente
d from static)
   TTimeStamp();

   // construction from timespec struct
   TTimeStamp(const timespec_t &ts) :
      fSec(Int_t(ts.tv_sec)), fNanoSec(ts.tv_nsec) { NormalizeNa
noSec(); }

   // construction from time_t and separate nsec
   TTimeStamp(time_t t, Int_t nsec) :
      fSec(Int_t(t)), fNanoSec(nsec) { NormalizeNanoSec(); }

   // construction from bits and pieces
   TTimeStamp(UInt_t year, UInt_t month,
              UInt_t day,  UInt_t hour,
              UInt_t min,  UInt_t sec,
              UInt_t nsec = 0, Bool_t isUTC = kTRUE, Int_t secOf
fset = 0);
```

```
   // compatibility with TDatime
   TTimeStamp(UInt_t date, UInt_t time, UInt_t nsec,
              Bool_t isUTC = kTRUE, Int_t secOffset = 0);

   // compatability with time() and DOS date
   TTimeStamp(UInt_t tloc, Bool_t isUTC = kTRUE, Int_t secOffset
 = 0,
              Bool_t dosDate = kFALSE);

   virtual ~TTimeStamp() { }

   // initialize to current time with nsec field incremented fro
m static
   void Set();

   // construction from bits and pieces
   void Set(Int_t year, Int_t month,  Int_t day,
            Int_t hour, Int_t min,    Int_t sec,
            Int_t nsec, Bool_t isUTC, Int_t secOffset);

   // compatibility with TDatime
   void Set(Int_t date,   Int_t time, Int_t nsec,
            Bool_t isUTC, Int_t secOffset);

   // compatability with time() and DOS date
   void Set(UInt_t tloc, Bool_t isUTC, Int_t secOffset, Bool_t d
osDate);

   // direct setters
   void SetSec(Int_t sec) { fSec = sec; }
   void SetNanoSec(Int_t nsec) { fNanoSec = nsec; }

   timespec_t   GetTimeSpec() const
      { timespec_t value = {fSec,fNanoSec}; return value; }
   time_t       GetSec() const { return fSec; }
   Int_t        GetNanoSec() const { return fNanoSec; }

   Double_t     AsDouble() const { return fSec + 1e-9 * fNanoSec
; }
```

```
   Double_t     AsJulianDate() const { return (AsDouble()/86400.0
 + 2440587.5); }

   // return stored time values converted to sidereal time
   Double_t     AsGMST(Double_t UT1Offset = 0 /*milliseconds*/)
const; //rval in hours
   Double_t     AsGAST(Double_t UT1Offset = 0 /*milliseconds*/)
const; //rval in hours
   Double_t     AsLMST(Double_t Longitude /*degrees*/, Double_t
UT1Offset = 0 /*milliseconds*/) const; //rval in hours
   Double_t     AsLAST(Double_t Longitude /*degrees*/, Double_t
UT1Offset = 0 /*milliseconds*/) const; //rval in hours

   const char  *AsString(const Option_t *option="") const;

   void         Copy(TTimeStamp &ts) const;
   UInt_t       GetDate(Bool_t inUTC = kTRUE, Int_t secOffset = 0
,
                        UInt_t *year = 0, UInt_t *month = 0,
                        UInt_t *day = 0) const;
   UInt_t       GetTime(Bool_t inUTC = kTRUE, Int_t secOffset = 0
,
                        UInt_t *hour = 0, UInt_t *min = 0,
                        UInt_t *sec = 0) const;
   Int_t        GetDayOfYear(Bool_t inUTC = kTRUE, Int_t secOffs
et = 0) const;
   Int_t        GetDayOfWeek(Bool_t inUTC = kTRUE, Int_t secOffs
et = 0) const;
   Int_t        GetMonth(Bool_t inUTC = kTRUE, Int_t secOffset =
0) const;
   Int_t        GetWeek(Bool_t inUTC = kTRUE, Int_t secOffset = 0
) const;
   Bool_t       IsLeapYear(Bool_t inUTC = kTRUE, Int_t secOffset
 = 0) const;

   void         Add(const TTimeStamp &offset);

   void         Print(const Option_t *option="") const;

   operator double() const { return AsDouble(); }
```

```
    // Utility functions
    static Int_t   GetZoneOffset();
    static time_t  MktimeFromUTC(tm_t *tmstruct);
    static void    DumpTMStruct(const tm_t &tmstruct);
    static Int_t   GetDayOfYear(Int_t day, Int_t month, Int_t yea
r);
    static Int_t   GetDayOfWeek(Int_t day, Int_t month, Int_t yea
r);
    static Int_t   GetWeek(Int_t day, Int_t month, Int_t year);
    static Bool_t  IsLeapYear(Int_t year);
```

code

example

到 4240

TTree

继承 TNamed, TAttLine, TAttFill, TAttMarker

A TTree object is a list of TBranch. To Create a TTree object one must:

- Create the TTree header via the TTree constructor
- Call the TBranch constructor for every branch.

To Fill this object, use member function Fill with no parameters. The Fill function loops on all defined TBranch.

class

```
   // Used as the max value for any TTree range operation.
   static constexpr Long64_t kMaxEntries = TVirtualTreePlayer::k
MaxEntries;

   // SetBranchAddress return values
   enum ESetBranchAddressStatus {
      kMissingBranch = -5,
      kInternalError = -4,
      kMissingCompiledCollectionProxy = -3,
      kMismatch = -2,
      kClassMismatch = -1,
      kMatch = 0,
      kMatchConversion = 1,
      kMatchConversionCollection = 2,
      kMakeClass = 3,
      kVoidPtr = 4,
      kNoCheck = 5
   };

   // TTree status bits
   enum {
      kForceRead   = BIT(11),
```

```
      kCircular    = BIT(12)
   };

   // Split level modifier
   enum {
      kSplitCollectionOfPointers = 100
   };

   class TClusterIterator
   {
   private:
      TTree    *fTree;        // TTree upon which we are iterati
ng.
      Int_t    fClusterRange; // Which cluster range are we look
ing at.
      Long64_t fStartEntry;   // Where does the cluster start.
      Long64_t fNextEntry;    // Where does the cluster end (exc
lusive).

      Long64_t GetEstimatedClusterSize();

   protected:
      friend class TTree;
      TClusterIterator(TTree *tree, Long64_t firstEntry);

   public:
      // Intentionally used the default copy constructor and def
ault destructor
      // as the TClusterIterator does not own the TTree.
      //  TClusterIterator(const TClusterIterator&);
      // ~TClusterIterator();

      // No public constructors, the iterator must be
      // created via TTree::GetClusterIterator

      // Move on to the next cluster and return the starting ent
ry
      // of this next cluster
      Long64_t Next();
```

```
      // Return the start entry of the current cluster.
      Long64_t GetStartEntry() {
         return fStartEntry;
      }

      // Return the first entry of the next cluster.
      Long64_t GetNextEntry() {
         return fNextEntry;
      }

      Long64_t operator()() { return Next(); }
   };

   TTree();
   TTree(const char* name, const char* title, Int_t splitlevel =
99);
   virtual ~TTree();

   virtual Int_t           AddBranchToCache(const char *bname, B
ool_t subbranches = kFALSE);
   virtual Int_t           AddBranchToCache(TBranch *branch,   B
ool_t subbranches = kFALSE);
   virtual Int_t           DropBranchFromCache(const char *bname
, Bool_t subbranches = kFALSE);
   virtual Int_t           DropBranchFromCache(TBranch *branch,
  Bool_t subbranches = kFALSE);
   virtual TFriendElement *AddFriend(const char* treename, const
char* filename = "");
   virtual TFriendElement *AddFriend(const char* treename, TFile
* file);
   virtual TFriendElement *AddFriend(TTree* tree, const char* al
ias = "", Bool_t warn = kFALSE);
   virtual void            AddTotBytes(Int_t tot) { fTotBytes +=
 tot; }
   virtual void            AddZipBytes(Int_t zip) { fZipBytes +=
 zip; }
   virtual Long64_t        AutoSave(Option_t* option = "");
// AutoSave tree header every fAutoSave bytes.
//   When large Trees are produced, it is safe to activate the A
utoSave
```

```
//   procedure. Some branches may have buffers holding many entr
ies.
//   AutoSave is automatically called by TTree::Fill when the nu
mber of bytes
//   generated since the previous AutoSave is greater than fAuto
Save bytes.
//   This function may also be invoked by the user, for example
every
//   N entries.
//   Each AutoSave generates a new key on the file.
//   Once the key with the tree header has been written, the pre
vious cycle
//   (if any) is deleted.
//   Note that calling TTree::AutoSave too frequently (or simila
rly calling
//   TTree::SetAutoSave with a small value) is an expensive oper
ation.
//   You should make tests for your own application to find a co
mpromise
//   between speed and the quantity of information you may loose
 in case of
//   a job crash.
//   In case your program crashes before closing the file holdin
g this tree,
//   the file will be automatically recovered when you will conn
ect the file
//   in UPDATE mode.
//   The Tree will be recovered at the status corresponding to t
he last AutoSave.
//   if option contains "SaveSelf", gDirectory->SaveSelf() is ca
lled.
//   This allows another process to analyze the Tree while the T
ree is being filled.
//   if option contains "FlushBaskets", TTree::FlushBaskets is c
alled and all
//   the current basket are closed-out and written to disk indiv
idually.
//   By default the previous header is deleted after having writ
ten the new header.
//   if option contains "Overwrite", the previous Tree header is
```

```
 deleted
//   before written the new header. This option is slightly fast
er, but
//   the default option is safer in case of a problem (disk quot
a exceeded)
//   when writing the new header.
//   The function returns the number of bytes written to the fil
e.
//   if the number of bytes is null, an error has occurred while
 writing
//   the header to the file.

   virtual Int_t           Branch(TCollection* list, Int_t bufsi
ze = 32000, Int_t splitlevel = 99, const char* name = "");
   virtual Int_t           Branch(TList* list, Int_t bufsize = 3
2000, Int_t splitlevel = 99);
   virtual Int_t           Branch(const char* folder, Int_t bufs
ize = 32000, Int_t splitlevel = 99);
   virtual TBranch        *Branch(const char* name, void* addres
s, const char* leaflist, Int_t bufsize = 32000);
           TBranch        *Branch(const char* name, char* addres
s, const char* leaflist, Int_t bufsize = 32000)
   {
      // Overload to avoid confusion between this signature and
the template instance.
      return Branch(name,(void*)address,leaflist,bufsize);
   }
   TBranch        *Branch(const char* name, Long_t address, const
 char* leaflist, Int_t bufsize = 32000)
   {
      // Overload to avoid confusion between this signature and
the template instance.
      return Branch(name,(void*)address,leaflist,bufsize);
   }
   TBranch        *Branch(const char* name, int address, const c
har* leaflist, Int_t bufsize = 32000)
   {
      // Overload to avoid confusion between this signature and
the template instance.
      return Branch(name,(void*)(Long_t)address,leaflist,bufsize
```

```
);
   }
#if !defined(__CINT__)
   virtual TBranch        *Branch(const char* name, const char*
classname, void* addobj, Int_t bufsize = 32000, Int_t splitlevel
 = 99);
#endif
   template <class T> TBranch *Branch(const char* name, const ch
ar* classname, T* obj, Int_t bufsize = 32000, Int_t splitlevel =
99)
   {
      // See BranchImpRed for details. Here we __ignore
      return BranchImpRef(name, classname, TBuffer::GetClass(typ
eid(T)), obj, bufsize, splitlevel);
   }
   template <class T> TBranch *Branch(const char* name, const ch
ar* classname, T** addobj, Int_t bufsize = 32000, Int_t splitlev
el = 99)
   {
      // See BranchImp for details
      return BranchImp(name, classname, TBuffer::GetClass(typeid
(T)), addobj, bufsize, splitlevel);
   }
   template <class T> TBranch *Branch(const char* name, T** addo
bj, Int_t bufsize = 32000, Int_t splitlevel = 99)
   {
      // See BranchImp for details
      return BranchImp(name, TBuffer::GetClass(typeid(T)), addob
j, bufsize, splitlevel);
   }
   template <class T> TBranch *Branch(const char* name, T* obj,
Int_t bufsize = 32000, Int_t splitlevel = 99)
   {
      // See BranchImp for details
      return BranchImpRef(name, TBuffer::GetClass(typeid(T)), TD
ataType::GetType(typeid(T)), obj, bufsize, splitlevel);
   }
   virtual TBranch        *Bronch(const char* name, const char*
classname, void* addobj, Int_t bufsize = 32000, Int_t splitlevel
 = 99);
```

```
   virtual TBranch        *BranchOld(const char* name, const char
* classname, void* addobj, Int_t bufsize = 32000, Int_t splitlev
el = 1);
   virtual TBranch        *BranchRef();
   virtual void            Browse(TBrowser*);
   virtual Int_t           BuildIndex(const char* majorname, con
st char* minorname = "0");
   TStreamerInfo          *BuildStreamerInfo(TClass* cl, void* p
ointer = 0, Bool_t canOptimize = kTRUE);
   virtual TFile          *ChangeFile(TFile* file);
   virtual TTree          *CloneTree(Long64_t nentries = -1, Opt
ion_t* option = "");
   virtual void            CopyAddresses(TTree*,Bool_t undo = kF
ALSE);
   virtual Long64_t        CopyEntries(TTree* tree, Long64_t nen
tries = -1, Option_t *option = "");
   virtual TTree          *CopyTree(const char* selection, Optio
n_t* option = "", Long64_t nentries = kMaxEntries, Long64_t firs
tentry = 0);
   virtual TBasket        *CreateBasket(TBranch*);
   virtual void            DirectoryAutoAdd(TDirectory *);
   Int_t                   Debug() const { return fDebug; }
   virtual void            Delete(Option_t* option = ""); // *ME
NU*
   virtual void            Draw(Option_t* opt) { Draw(opt, "", ""
, kMaxEntries, 0); }
   virtual Long64_t        Draw(const char* varexp, const TCut&
selection, Option_t* option = "", Long64_t nentries = kMaxEntrie
s, Long64_t firstentry = 0);
   virtual Long64_t        Draw(const char* varexp, const char*
selection, Option_t* option = "", Long64_t nentries = kMaxEntrie
s, Long64_t firstentry = 0); // *MENU*
   virtual void            DropBaskets();/// Remove some baskets
 from memory.
   virtual void            DropBuffers(Int_t nbytes);/// Drop br
anch buffers to accommodate nbytes below MaxVirtualsize.
   virtual Int_t           Fill();//填充到buffer中，一定数量之后写入
硬盘
/// Fill all branches.
/// This function loops on all the branches of this tree.  For
```

```
/// each branch, it copies to the branch buffer (basket) the cur
rent
/// values of the leaves data types. If a leaf is a simple data
type,
/// a simple conversion to a machine independent format has to b
e done.
/// This machine independent version of the data is copied into a

/// basket (each branch has its own basket).  When a basket is f
ull
/// (32k worth of data by default), it is then optionally compre
ssed
/// and written to disk (this operation is also called committin
g or
/// 'flushing' the basket).  The committed baskets are then
/// immediately removed from memory.
/// The function returns the number of bytes committed to the
/// individual branches.
/// If a write error occurs, the number of bytes returned is -1.
/// If no data are written, because, e.g., the branch is disable
d,
/// the number of bytes returned is 0.
/// __The baskets are flushed and the Tree header saved at regul
ar intervals__
/// At regular intervals, when the amount of data written so far
 is
/// greater than fAutoFlush (see SetAutoFlush) all the baskets a
re flushed to disk.
/// This makes future reading faster as it guarantees that baske
ts belonging to nearby
/// entries will be on the same disk region.
/// When the first call to flush the baskets happen, we also tak
e this opportunity
/// to optimize the baskets buffers.
/// We also check if the amount of data written is greater than
fAutoSave (see SetAutoSave).
/// In this case we also write the Tree header. This makes the T
ree recoverable up to this point
/// in case the program writing the Tree crashes.
/// The decisions to FlushBaskets and Auto Save can be made base
```

```
d either on the number
/// of bytes written (fAutoFlush and fAutoSave negative) or on t
he number of entries
/// written (fAutoFlush and fAutoSave positive).
/// Note that the user can decide to call FlushBaskets and AutoS
ave in her event loop
/// base on the number of events written instead of the number o
f bytes written.
/// Note that calling FlushBaskets too often increases the IO ti
me.
/// Note that calling AutoSave too often increases the IO time a
nd also the file size.

   virtual TBranch        *FindBranch(const char* name);
/// Return the branch that correspond to the path 'branchname',
which can
/// include the name of the tree or the omitted name of the pare
nt branches.
/// In case of ambiguity, returns the first match.

   virtual TLeaf          *FindLeaf(const char* name);/// Find l
eaf..
   virtual Int_t           Fit(const char* funcname, const char*
 varexp, const char* selection = "", Option_t* option = "", Opti
on_t* goption = "", Long64_t nentries = kMaxEntries, Long64_t fi
rstentry = 0); // *MENU*
/// Fit  a projected item(s) from a tree.
/// funcname is a TF1 function.
/// See TTree::Draw() for explanations of the other parameters.
/// By default the temporary histogram created is called htemp.
/// If varexp contains >>hnew , the new histogram created is cal
led hnew
/// and it is kept in the current directory.
/// The function returns the number of selected entries.
/// ## Return status
///  The function returns the status of the histogram fit (see T
H1::Fit)
///  If no entries were selected, the function returns -1;
///   (i.e. fitResult is null is the fit is OK)
```

```
   virtual Int_t           FlushBaskets() const;
/// Write to disk all the basket that have not yet been individu
ally written.
/// Return the number of bytes written or -1 in case of write er
ror.

   virtual const char     *GetAlias(const char* aliasName) const;
/// Returns the expanded value of the alias.  Search in the frie
nds if any.
   virtual Long64_t        GetAutoFlush() const {return fAutoFlu
sh;}
   virtual Long64_t        GetAutoSave()  const {return fAutoSav
e;}
   virtual TBranch        *GetBranch(const char* name);/// Retur
n pointer to the branch with the given name in this tree or its
friends.
   virtual TBranchRef     *GetBranchRef() const { return fBranch
Ref; };
   virtual Bool_t          GetBranchStatus(const char* branchnam
e) const;
/// Return status of branch with name branchname.
/// - 0 if branch is not activated
/// - 1 if branch is activated

   static  Int_t           GetBranchStyle();
/// Static function returning the current branch style.
/// - style = 0 old Branch
/// - style = 1 new Bronch

   virtual Long64_t        GetCacheSize() const { return fCacheS
ize; }
   virtual TClusterIterator GetClusterIterator(Long64_t firstent
ry);
/// Return an iterator over the cluster of baskets starting at f
irstentry.
/// This iterator is not yet supported for TChain object.

   virtual Long64_t        GetChainEntryNumber(Long64_t entry) c
onst { return entry; }
   virtual Long64_t        GetChainOffset() const { return fChai
```

```
nOffset; }
   TFile                   *GetCurrentFile() const;/// Return poi
nter to the current file.
           Int_t           GetDefaultEntryOffsetLen() const {ret
urn fDefaultEntryOffsetLen;}
           Long64_t        GetDebugMax()  const { return fDebugM
ax; }
           Long64_t        GetDebugMin()  const { return fDebugM
in; }
   TDirectory              *GetDirectory() const { return fDirect
ory; }
   virtual Long64_t        GetEntries() const   { return fEntrie
s; }//获取entry数
   virtual Long64_t        GetEntries(const char *selection);
/// Return the number of entries matching the selection.
/// Return -1 in case of errors.
/// If the selection uses any arrays or containers, we return th
e number
/// of entries where at least one element match the selection.
/// GetEntries is implemented using the selector class TSelector
Entries,
/// which can be used directly (see code in TTreePlayer::GetEntr
ies) for
/// additional option.
/// If SetEventList was used on the TTree or TChain, only that s
ubset
/// of entries will be considered.

   virtual Long64_t        GetEntriesFast() const   { return fEn
tries; }
   virtual Long64_t        GetEntriesFriend() const;
/// Return pointer to the 1st Leaf named name in any Branch of t
his Tree or
/// any branch in the list of friend trees.

   virtual Long64_t        GetEstimate() const { return fEstimat
e; }
   virtual Int_t           GetEntry(Long64_t entry = 0, Int_t ge
tall = 0);
/// Read all branches of entry and return total number of bytes
```

```
read.
/// - getall = 0 : get only active branches
/// - getall = 1 : get all branches
/// The function returns the number of bytes read from the input
 buffer.
/// If entry does not exist the function returns 0.
/// If an I/O error occurs, the function returns -1.
/// If the Tree has friends, also read the friends entry.
/// ## IMPORTANT NOTE
///
/// By default, GetEntry reuses the space allocated by the previ
ous object
/// for each branch. You can force the previous object to be aut
omatically
/// deleted if you call mybranch.SetAutoDelete(kTRUE) (default i
s kFALSE).

           Int_t           GetEvent(Long64_t entry = 0, Int_t ge
tall = 0) { return GetEntry(entry, getall); }
   virtual Int_t           GetEntryWithIndex(Int_t major, Int_t
minor = 0);
/// Read entry corresponding to major and minor number.
///  The function returns the total number of bytes read.
///  If the Tree has friend trees, the corresponding entry with
///  the index values (major,minor) is read. Note that the maste
r Tree
///  and its friend may have different entry serial numbers corr
esponding
///  to (major,minor).

   virtual Long64_t        GetEntryNumberWithBestIndex(Long64_t
major, Long64_t minor = 0) const;
/// Return entry number corresponding to major and minor number.
/// Note that this function returns only the entry number, not t
he data
/// To read the data corresponding to an entry number, use TTree
::GetEntryWithIndex
/// the BuildIndex function has created a table of Long64_t* of
sorted values
/// corresponding to val = major<<31 + minor;
```

```
/// The function performs binary search in this sorted table.
/// If it finds a pair that matches val, it returns directly the
/// index in the table.
/// If an entry corresponding to major and minor is not found, t
he function
/// returns the index of the major,minor pair immediately lower
than the
/// requested value, ie it will return -1 if the pair is lower t
han
/// the first entry in the index.
/// See also GetEntryNumberWithIndex

   virtual Long64_t        GetEntryNumberWithIndex(Long64_t majo
r, Long64_t minor = 0) const;
/// Return entry number corresponding to major and minor number.
/// Note that this function returns only the entry number, not t
he data
/// To read the data corresponding to an entry number, use TTree
::GetEntryWithIndex
/// the BuildIndex function has created a table of Long64_t* of
sorted values
/// corresponding to val = major<<31 + minor;
/// The function performs binary search in this sorted table.
/// If it finds a pair that matches val, it returns directly the
/// index in the table, otherwise it returns -1.
/// See also GetEntryNumberWithBestIndex

   TEventList             *GetEventList() const { return fEventL
ist; }
   virtual TEntryList     *GetEntryList();///Returns the entry l
ist, set to this tree
   virtual Long64_t        GetEntryNumber(Long64_t entry) const;
/// Return entry number corresponding to entry.
/// if no TEntryList set returns entry
/// else returns the entry number corresponding to the list inde
x=entry

   virtual Int_t           GetFileNumber() const { return fFileN
umber; }
   virtual TTree          *GetFriend(const char*) const;/// Retu
```

```
rn a pointer to the TTree friend whose name or alias is 'friendn
ame.
   virtual const char     *GetFriendAlias(TTree*) const;
/// If the 'tree' is a friend, this method returns its alias nam
e.
/// This alias is an alternate name for the tree.
/// It can be used in conjunction with a branch or leaf name in
a TTreeFormula,
/// to specify in which particular tree the branch or leaf can b
e found if
/// the friend trees have branches or leaves with the same name
as the master
/// tree.
/// It can also be used in conjunction with an alias created usi
ng
/// TTree::SetAlias in a TTreeFormula, e.g.:
///      maintree->Draw("treealias.fPx - treealias.myAlias");
/// where fPx is a branch of the friend tree aliased as 'treeali
as' and 'myAlias'
/// was created using TTree::SetAlias on the friend tree.
/// However, note that 'treealias.myAlias' will be expanded lite
rally,
/// without remembering that it comes from the aliased friend an
d thus
/// the branch name might not be disambiguated properly, which m
eans
/// that you may not be able to take advantage of this feature.

   TH1                    *GetHistogram() { return GetPlayer()->
GetHistogram(); }
   virtual Int_t          *GetIndex() { return &fIndex.fArray[0]
; }
   virtual Double_t       *GetIndexValues() { return &fIndexValu
es.fArray[0]; }
   virtual TIterator      *GetIteratorOnAllLeaves(Bool_t dir = k
IterForward);
/// Creates a new iterator that will go through all the leaves o
n the tree itself and its friend.

   virtual TLeaf          *GetLeaf(const char* branchname, const
```

```
char* leafname);
/// Return pointer to the 1st Leaf named name in any Branch of t
his
/// Tree or any branch in the list of friend trees.
/// The leaf name can contain the name of a friend tree with the
/// syntax: friend_dir_and_tree.full_leaf_name
/// the friend_dir_and_tree can be of the form:
///     TDirectoryName/TreeName

   virtual TLeaf          *GetLeaf(const char* name);
/// Return pointer to the 1st Leaf named name in any Branch of t
his
/// Tree or any branch in the list of friend trees.
/// aname may be of the form branchname/leafname

   virtual TList          *GetListOfClones() { return fClones; }
   virtual TObjArray      *GetListOfBranches() { return &fBranch
es; }
   virtual TObjArray      *GetListOfLeaves() { return &fLeaves;
}
   virtual TList          *GetListOfFriends() const { return fFr
iends; }
   virtual TList          *GetListOfAliases() const { return fAl
iases; }

   // GetMakeClass is left non-virtual for efficiency reason.
   // Making it virtual affects the performance of the I/O
           Int_t           GetMakeClass() const { return fMakeCl
ass; }

   virtual Long64_t        GetMaxEntryLoop() const { return fMax
EntryLoop; }
   virtual Double_t        GetMaximum(const char* columname);
/// Return maximum of column with name columname.
/// if the Tree has an associated TEventList or TEntryList, the
maximum
/// is computed for the entries in this list.

   static  Long64_t        GetMaxTreeSize();/// Static function
which returns the tree file size limit in bytes.
```

```
   virtual Long64_t        GetMaxVirtualSize() const { return fM
axVirtualSize; }
   virtual Double_t        GetMinimum(const char* columname);
/// Return minimum of column with name columname.
/// if the Tree has an associated TEventList or TEntryList, the
minimum
/// is computed for the entries in this list.

   virtual Int_t           GetNbranches() { return fBranches.Get
EntriesFast(); }
   TObject                *GetNotify() const { return fNotify; }
   TVirtualTreePlayer     *GetPlayer();/// Load the TTreePlayer
(if not already done).
   virtual Int_t           GetPacketSize() const { return fPacke
tSize; }
   virtual TVirtualPerfStats *GetPerfStats() const { return fPer
fStats; }
   virtual Long64_t        GetReadEntry()  const { return fReadE
ntry; }
   virtual Long64_t        GetReadEvent()  const { return fReadE
ntry; }
   virtual Int_t           GetScanField()  const { return fScanF
ield; }
   TTreeFormula           *GetSelect()    { return GetPlayer()->
GetSelect(); }
   virtual Long64_t        GetSelectedRows() { return GetPlayer(
)->GetSelectedRows(); }
   virtual Int_t           GetTimerInterval() const { return fTi
merInterval; }
           TBuffer*        GetTransientBuffer(Int_t size);
   virtual Long64_t        GetTotBytes() const { return fTotByte
s; }
   virtual TTree          *GetTree() const { return const_cast<T
Tree*>(this); }
   virtual TVirtualIndex  *GetTreeIndex() const { return fTreeIn
dex; }
   virtual Int_t           GetTreeNumber() const { return 0; }
   virtual Int_t           GetUpdate() const { return fUpdate; }
   virtual TList          *GetUserInfo();
/// Return a pointer to the list containing user objects associa
```

```
ted to this tree.
/// The list is automatically created if it does not exist.
/// WARNING: By default the TTree destructor will delete all obj
ects added
/// to this list. If you do not want these objects to be deleted,

/// call:
///     mytree->GetUserInfo()->Clear();
/// before deleting the tree.

   // See TSelectorDraw::GetVar
   TTreeFormula           *GetVar(Int_t i)  { return GetPlayer()
->GetVar(i); }
   // See TSelectorDraw::GetVar
   TTreeFormula           *GetVar1() { return GetPlayer()->GetVa
r1(); }
   // See TSelectorDraw::GetVar
   TTreeFormula           *GetVar2() { return GetPlayer()->GetVa
r2(); }
   // See TSelectorDraw::GetVar
   TTreeFormula           *GetVar3() { return GetPlayer()->GetVa
r3(); }
   // See TSelectorDraw::GetVar
   TTreeFormula           *GetVar4() { return GetPlayer()->GetVa
r4(); }
   // See TSelectorDraw::GetVal
   virtual Double_t       *GetVal(Int_t i)   { return GetPlayer(
)->GetVal(i); }
   // See TSelectorDraw::GetVal
   virtual Double_t       *GetV1()   { return GetPlayer()->GetV1
(); }
   // See TSelectorDraw::GetVal
   virtual Double_t       *GetV2()   { return GetPlayer()->GetV2
(); }
   // See TSelectorDraw::GetVal
   virtual Double_t       *GetV3()   { return GetPlayer()->GetV3
(); }
   // See TSelectorDraw::GetVal
   virtual Double_t       *GetV4()   { return GetPlayer()->GetV4
(); }
```

```
   virtual Double_t       *GetW()    { return GetPlayer()->GetW(
); }
   virtual Double_t        GetWeight() const   { return fWeight;
 }
   virtual Long64_t        GetZipBytes() const { return fZipByte
s; }
   virtual void            IncrementTotalBuffers(Int_t nbytes) {
 fTotalBuffers += nbytes; }
   Bool_t                  IsFolder() const { return kTRUE; }
   virtual Int_t           LoadBaskets(Long64_t maxmemory = 2000
000000);
/// Read in memory all baskets from all branches up to the limit
 of maxmemory bytes.
/// If maxmemory is non null and positive SetMaxVirtualSize is c
alled
/// with this value. Default for maxmemory is 2000000000 (2 Giga
bytes).
/// The function returns the total number of baskets read into m
emory
/// if negative an error occurred while loading the branches.
/// This method may be called to force branch baskets in memory
/// when random access to branch entries is required.
/// If random access to only a few branches is required, you sho
uld
/// call directly TBranch::LoadBaskets.

   virtual Long64_t        LoadTree(Long64_t entry);
/// Set current entry.
/// Returns -2 if entry does not exist (just as TChain::LoadTree
()).
/// Note: This function is overloaded in TChain.

   virtual Long64_t        LoadTreeFriend(Long64_t entry, TTree*
 T);
/// Load entry on behalf of our master tree, we may use an index.

/// Called by LoadTree() when the masterTree looks for the entry
/// number in a friend tree (us) corresponding to the passed ent
ry
/// number in the masterTree.
```

```
/// If we have no index, our entry number and the masterTree ent
ry
/// number are the same.
/// If we *do* have an index, we must find the (major, minor) va
lue pair
/// in masterTree to locate our corresponding entry.

   virtual Int_t           MakeClass(const char* classname = 0,
Option_t* option = "");
/// Generate a skeleton analysis class for this tree.
/// The following files are produced: classname.h and classname.
C.
/// If classname is 0, classname will be called "nameoftree".
/// The generated code in classname.h includes the following:
/// - Identification of the original tree and the input file nam
e.
/// - Definition of an analysis class (data members and member f
unctions).
/// - The following member functions:
///   - constructor (by default opening the tree file),
///   - GetEntry(Long64_t entry),
///   - Init(TTree* tree) to initialize a new TTree,
///   - Show(Long64_t entry) to read and dump entry.
/// The generated code in classname.C includes only the main
/// analysis function Loop.
/// NOTE: Do not use the code generated for a single TTree which
 is part
/// of a TChain to process that entire TChain.  The maximum dime
nsions
/// calculated for arrays on the basis of a single TTree from th
e TChain
/// might be (will be!) too small when processing all of the TTr
ees in
/// the TChain.  You must use myChain.MakeClass() to generate th
e code,
/// not myTree.MakeClass(...).

   virtual Int_t           MakeCode(const char* filename = 0);
/// Generate a skeleton function for this tree.
/// The function code is written on filename.
```

```
/// If filename is 0, filename will be called nameoftree.C
/// The generated code includes the following:
/// - Identification of the original Tree and Input file name,
/// - Opening the Tree file,
/// - Declaration of Tree variables,
/// - Setting of branches addresses,
/// - A skeleton for the entry loop.
/// To use this function:
/// - Open your Tree file (eg: TFile f("myfile.root");)
/// - T->MakeCode("MyAnalysis.C");
/// where T is the name of the TTree in file myfile.root
/// and MyAnalysis.C the name of the file created by this functi
on.
/// NOTE: Since the implementation of this function, a new and b
etter
/// function TTree::MakeClass() has been developed.

   virtual Int_t           MakeProxy(const char* classname, const
 char* macrofilename = 0, const char* cutfilename = 0, const char
* option = 0, Int_t maxUnrolling = 3);
/// Generate a skeleton analysis class for this Tree using TBran
chProxy.
/// TBranchProxy is the base of a class hierarchy implementing an

/// indirect access to the content of the branches of a TTree.
/// "proxyClassname" is expected to be of the form:
///     [path/]fileprefix
/// The skeleton will then be generated in the file:
///     fileprefix.h
/// located in the current directory or in 'path/' if it is spec
ified.
/// The class generated will be named 'fileprefix'
///
/// "macrofilename" and optionally "cutfilename" are expected to
 point
/// to source files which will be included by the generated skel
eton.
/// Method of the same name as the file(minus the extension and
path)
/// will be called by the generated skeleton's Process method as
```

```
 follow:
///     [if (cutfilename())] htemp->Fill(macrofilename());
/// "option" can be used select some of the optional features du
ring
/// the code generation.  The possible options are:
/// - nohist : indicates that the generated ProcessFill should n
ot fill the histogram.
/// 'maxUnrolling' controls how deep in the class hierarchy does
 the
/// system 'unroll' classes that are not split.  Unrolling a cla
ss
/// allows direct access to its data members (this emulates the
behavior
/// of TTreeFormula).
///
/// The main features of this skeleton are:
/// * on-demand loading of branches
/// * ability to use the 'branchname' as if it was a data member
/// * protection against array out-of-bounds errors
/// * ability to use the branch data as an object (when the user
 code is available)
/// If a file name macrofilename.h (or .hh, .hpp, .hxx, .hPP, .h
XX) exist
/// it is included before the declaration of the proxy class.  T
his can
/// be used in particular to insure that the include files neede
d by
/// the macro file are properly loaded.
/// The default histogram is accessible via the variable named '
htemp'.
/// If the library of the classes describing the data in the bra
nch is
/// loaded, the skeleton will add the needed #include statements
 and
/// give the ability to access the object stored in the branches.


   virtual Int_t           MakeSelector(const char* selector = 0
, Option_t* option = "");//生成要Process()的文件
/// Generate skeleton selector class for this tree.
```

```
/// The following files are produced: selector.h and selector.C.
/// If selector is 0, the selector will be called "nameoftree".
/// The option can be used to specify the branches that will hav
e a data member.
///    - If option is "=legacy", a pre-ROOT6 selector will be ge
nerated (data
///      members and branch pointers instead of TTreeReaders).
///    - If option is empty, readers will be generated for each
leaf.
///    - If option is "@", readers will be generated for the top
most branches.
///    - Individual branches can also be picked by their name:
///       - "X" generates readers for leaves of X.
///       - "@X" generates a reader for X as a whole.
///       - "@X;Y" generates a reader for X as a whole and also
readers for the
///         leaves of Y.
///    - For further examples see the figure below.
/// The generated code in selector.h includes the following:
///    - Identification of the original Tree and Input file name
///    - Definition of selector class (data and functions)
///    - The following class functions:
///       - constructor and destructor
///       - void    Begin(TTree *tree)
///       - void    SlaveBegin(TTree *tree)
///       - void    Init(TTree *tree)
///       - Bool_t  Notify()
///       - Bool_t  Process(Long64_t entry)
///       - void    Terminate()
///       - void    SlaveTerminate()
/// The class selector derives from TSelector.
/// The generated code in selector.C includes empty functions de
fined above.
/// To use this function:
///    - connect your Tree file (eg: `TFile f("myfile.root");`)
///    - `T->MakeSelector("myselect");`
/// where T is the name of the Tree in file myfile.root
/// and myselect.h, myselect.C the name of the files created by
this function.
/// In a ROOT session, you can do:
```

```
///     root > T->Process("myselect.C")

   Bool_t                  MemoryFull(Int_t nbytes);/// Check if
 adding nbytes to memory we are still below MaxVirtualsize.
   virtual Long64_t        Merge(TCollection* list, Option_t* op
tion = "");
/// Merge the trees in the TList into this tree.
/// Returns the total number of entries in the merged tree.

   virtual Long64_t        Merge(TCollection* list, TFileMergeIn
fo *info);
/// Merge the trees in the TList into this tree.
/// If info->fIsFirst is true, first we clone this TTree info th
e directory
/// info->fOutputDirectory and then overlay the new TTree inform
ation onto
/// this TTree object (so that this TTree object is now the appr
opriate to
/// use for further merging).
/// Returns the total number of entries in the merged tree.

   static  TTree          *MergeTrees(TList* list, Option_t* opt
ion = "");
/// Static function merging the trees in the TList into a new tr
ee.
/// Trees in the list can be memory or disk-resident trees.
/// The new tree is created in the current directory (memory if
gROOT).

   virtual Bool_t          Notify();/// Function called when loa
ding a new class library.
   virtual void            OptimizeBaskets(ULong64_t maxMemory=1
0000000, Float_t minComp=1.1, Option_t *option="");
/// This function may be called after having filled some entries
 in a Tree
/// Using the information in the existing branch buffers, it wil
l reassign
/// new branch buffer sizes to optimize time and memory.
/// The function computes the best values for branch buffer size
s such that
```

```
/// the total buffer sizes is less than maxMemory and nearby ent
ries written
/// at the same time.
/// In case the branch compression factor for the data written s
o far is less
/// than compMin, the compression is disabled.
/// if option ="d" an analysis report is printed.

   TPrincipal             *Principal(const char* varexp = "", co
nst char* selection = "", Option_t* option = "np", Long64_t nent
ries = kMaxEntries, Long64_t firstentry = 0);
/// Interface to the Principal Components Analysis class.
/// Create an instance of TPrincipal
/// Fill it with the selected variables
/// - if option "n" is specified, the TPrincipal object is fille
d with
///                 normalized variables.
/// - If option "p" is specified, compute the principal componen
ts
/// - If option "p" and "d" print results of analysis
/// - If option "p" and "h" generate standard histograms
/// - If option "p" and "c" generate code of conversion functions

/// - return a pointer to the TPrincipal object. It is the user
responsibility
/// - to delete this object.
/// - The option default value is "np"
/// see TTree::Draw for explanation of the other parameters.
/// The created object is  named "principal" and a reference to
it
/// is added to the list of specials Root objects.

   virtual void            Print(Option_t* option = "") const; /
/ *MENU*
/// Print a summary of the tree contents.
/// -  If option contains "all" friend trees are also printed.
/// -  If option contains "toponly" only the top level branches
are printed.
/// -  If option contains "clusters" information about the clust
er of baskets is printed.
```

```
/// Wildcarding can be used to print only a subset of the branch
es, e.g.,
/// T.Print("Elec*") will print all branches with name starting
with "Elec".

   virtual void            PrintCacheStats(Option_t* option = "")
 const;
/// print statistics about the TreeCache for this tree, like
/// if option = "a" the list of blocks in the cache is printed

   virtual Long64_t        Process(const char* filename, Option_
t* option = "", Long64_t nentries = kMaxEntries, Long64_t firste
ntry = 0); // *MENU*
/// Process this tree executing the TSelector code in the specif
ied filename.
/// The return value is -1 in case of error and TSelector::GetSt
atus() in
/// in case of success.
/// The code in filename is loaded (interpreted or compiled, see
 below),
/// filename must contain a valid class implementation derived f
rom TSelector,
/// where TSelector has the following member functions:
/// - `Begin()`:         called every time a loop on the tree st
arts,
///                      a convenient place to create your histo
grams.
/// - `SlaveBegin()`:    called after Begin(), when on PROOF cal
led only on the
///                      slave servers.
/// - `Process()`:       called for each event, in this function
 you decide what
///                      to read and fill your histograms.
/// - `SlaveTerminate`:  called at the end of the loop on the tr
ee, when on PROOF
///                      called only on the slave servers.
/// - `Terminate()`:     called at the end of the loop on the tr
ee,
///                      a convenient place to draw/fit your his
tograms.
```

```
/// If filename is of the form file.C, the file will be interpre
ted.
/// If filename is of the form file.C++, the file file.C will be
 compiled
/// and dynamically loaded.
/// If filename is of the form file.C+, the file file.C will be
compiled
/// and dynamically loaded. At next call, if file.C is older tha
n file.o
/// and file.so, the file.C is not compiled, only file.so is loa
ded.

#if defined(__CINT__)
#if defined(R__MANUAL_DICT)
   virtual Long64_t        Process(void* selector, Option_t* opt
ion = "", Long64_t nentries = kMaxEntries, Long64_t firstentry =
0);
#endif
#else
   virtual Long64_t        Process(TSelector* selector, Option_t
* option = "", Long64_t nentries = kMaxEntries, Long64_t firsten
try = 0);
/// Process this tree executing the code in the specified select
or.
/// The return value is -1 in case of error and TSelector::GetSt
atus() in
/// in case of success.
///   The TSelector class has the following member functions:
/// - `Begin()`:        called every time a loop on the tree sta
rts,
///                     a convenient place to create your histog
rams.
/// - `SlaveBegin()`:   called after Begin(), when on PROOF call
ed only on the
///                     slave servers.
/// - `Process()`:      called for each event, in this function
you decide what
///                     to read and fill your histograms.
/// - `SlaveTerminate`: called at the end of the loop on the tre
e, when on PROOF
```

```
///                      called only on the slave servers.
/// - `Terminate()`:     called at the end of the loop on the tre
e,
///                      a convenient place to draw/fit your hist
ograms.
///  If the Tree (Chain) has an associated EventList, the loop i
s on the nentries
///  of the EventList, starting at firstentry, otherwise the loo
p is on the
///  specified Tree entries.
#endif
   virtual Long64_t        Project(const char* hname, const char
* varexp, const char* selection = "", Option_t* option = "", Lon
g64_t nentries = kMaxEntries, Long64_t firstentry = 0);
/// Make a projection of a tree using selections.
/// Depending on the value of varexp (described in Draw) a 1-D,
2-D, etc.,
/// projection of the tree will be filled in histogram hname.
/// Note that the dimension of hname must match with the dimensi
on of varexp.

   virtual TSQLResult     *Query(const char* varexp = "", const
char* selection = "", Option_t* option = "", Long64_t nentries =
 kMaxEntries, Long64_t firstentry = 0);/// Loop over entries and
 return a TSQLResult object containing entries following selecti
on.
   virtual Long64_t        ReadFile(const char* filename, const
char* branchDescriptor = "", char delimiter = ' ');
/// Create or simply read branches from filename.
/// if branchDescriptor = "" (default), it is assumed that the T
ree descriptor
/// is given in the first line of the file with a syntax like
///     A/D:Table[2]/F:Ntracks/I:astring/C
/// otherwise branchDescriptor must be specified with the above
syntax.
/// - If the type of the first variable is not specified, it is
assumed to be "/F"
/// - If the type of any other variable is not specified, the ty
pe of the previous
///   variable is assumed. eg
```

```
///     - `x:y:z`      (all variables are assumed of type "F"
///     - `x/D:y:z`    (all variables are of type "D"
///     - `x:y/D:z`    (x is type "F", y and z of type "D"
/// delimiter allows for the use of another delimiter besides wh
itespace.
/// This provides support for direct import of common data file
formats
/// like csv.  If delimiter != ' ' and branchDescriptor == "", t
hen the
/// branch description is taken from the first line in the file,
 but
/// delimiter is used for the branch names tokenization rather t
han ':'.
/// Note however that if the values in the first line do not use
 the
/// /[type] syntax, all variables are assumed to be of type "F".
/// If the filename ends with extensions .csv or .CSV and a deli
miter is
/// not specified (besides ' '), the delimiter is automatically
set to ','.
/// Lines in the input file starting with "#" are ignored. Leadi
ng whitespace
/// for each column data is skipped. Empty lines are skipped.
/// A TBranch object is created for each variable in the express
ion.
/// The total number of rows read from the file is returned.

   virtual Long64_t        ReadStream(std::istream& inputStream,
const char* branchDescriptor = "", char delimiter = ' ');
/// Create or simply read branches from an input stream.
/// See reference information for TTree::ReadFile

   virtual void            Refresh();
///  Refresh contents of this tree and its branches from the cur
rent status on disk.
///  One can call this function in case the tree file is being
///  updated by another process.

   virtual void            RecursiveRemove(TObject *obj);
/// Make sure that obj (which is being deleted or will soon be)
```

```
is no
/// longer referenced by this TTree.

   virtual void            RemoveFriend(TTree*);/// Remove a fri
end from the list of friends.
   virtual void            Reset(Option_t* option = "");// Reset
 baskets, buffers and entries count in all branches and leaves.
   virtual void            ResetAfterMerge(TFileMergeInfo *);///
 Resets the state of this TTree after a merge (keep the customiz
ation but forget the data).
   virtual void            ResetBranchAddress(TBranch *);
/// Tell all of our branches to set their addresses to zero.
/// Note: If any of our branches own any objects, they are delet
ed.

   virtual void            ResetBranchAddresses();/// Tell all o
f our branches to drop their current objects and allocate new on
es.
   virtual Long64_t        Scan(const char* varexp = "", const c
har* selection = "", Option_t* option = "", Long64_t nentries =
kMaxEntries, Long64_t firstentry = 0); // *MENU*
// Loop over tree entries and print entries passing selection.
// If varexp is 0 (or "") then print only first 8 columns.
// If varexp = "*" print all columns.
// Otherwise a columns selection can be made using "var1:var2:va
r3".
// See TTreePlayer::Scan for more information

   virtual Bool_t          SetAlias(const char* aliasName, const
char* aliasFormula);
/// Set a tree variable alias.
/// Set an alias for an expression/formula based on the tree 'va
riables'.
/// The content of 'aliasName' can be used in TTreeFormula (i.e.
 TTree::Draw,
/// TTree::Scan, TTreeViewer) and will be evaluated as the conte
nt of
/// 'aliasFormula'.
/// If the content of 'aliasFormula' only contains symbol names,
 periods and
```

```
/// array index specification (for example event.fTracks[3]), th
en
/// the content of 'aliasName' can be used as the start of symbo
l.
/// If the alias 'aliasName' already existed, it is replaced by
the new value.
/// When being used, the alias can be preceded by an eventual 'F
riend Alias'
/// (see TTree::GetFriendAlias)
/// Return true if it was added properly.

   virtual void            SetAutoSave(Long64_t autos = -3000000
00);
/// This function may be called at the start of a program to cha
nge
/// the default value for fAutoSave (and for SetAutoSave) is -30
0000000, ie 300 MBytes
/// When filling the Tree the branch buffers as well as the Tree
 header
/// will be flushed to disk when the watermark is reached.
/// If fAutoSave is positive the watermark is reached when a mul
tiple of fAutoSave
/// entries have been written.
/// If fAutoSave is negative the watermark is reached when -fAut
oSave bytes
/// have been written to the file.
/// In case of a program crash, it will be possible to recover t
he data in the Tree
/// up to the last AutoSave point.

   virtual void            SetAutoFlush(Long64_t autof = -300000
00);
/// This function may be called at the start of a program to cha
nge
/// the default value for fAutoFlush.
/// ### CASE 1 : autof > 0
/// autof is the number of consecutive entries after which TTree
::Fill will
/// flush all branch buffers to disk.
/// ### CASE 2 : autof < 0
```

```
/// When filling the Tree the branch buffers will be flushed to
disk when
/// more than autof bytes have been written to the file. At the
first FlushBaskets
/// TTree::Fill will replace fAutoFlush by the current value of
fEntries.
/// Calling this function with autof<0 is interesting when it is
 hard to estimate
/// the size of one entry. This value is also independent of the
 Tree.
/// The Tree is initialized with fAutoFlush=-30000000, ie that,
by default,
/// the first AutoFlush will be done when 30 MBytes of data are
written to the file.
/// ### CASE 3 : autof = 0
/// The AutoFlush mechanism is disabled.
/// Flushing the buffers at regular intervals optimize the locat
ion of
/// consecutive entries on the disk by creating clusters of bask
ets.
/// A cluster of baskets is a set of baskets that contains all
/// the data for a (consecutive) set of entries and that is stor
ed
/// consecutively on the disk.   When reading all the branches,
this
/// is the minimum set of baskets that the TTreeCache will read.

   virtual void            SetBasketSize(const char* bname, Int_
t buffsize = 16000);
/// Set a branch's basket size.
/// bname is the name of a branch.
/// - if bname="*", apply to all branches.
/// - if bname="xxx*", apply to all branches with name starting
with xxx
/// see TRegexp for wildcarding options
/// buffsize = branc basket size

#if !defined(__CINT__)
   virtual Int_t           SetBranchAddress(const char *bname,vo
id *add, TBranch **ptr = 0);
```

```
/// Change branch address, dealing with clone trees properly.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.
/// Note: See the comments in TBranchElement::SetAddress() for t
he
/// meaning of the addr parameter and the object ownership polic
y.
#endif
   virtual Int_t           SetBranchAddress(const char *bname,vo
id *add, TClass *realClass, EDataType datatype, Bool_t isptr);
/// Verify the validity of the type of addr before calling SetBr
anchAddress.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.
/// Note: See the comments in TBranchElement::SetAddress() for t
he
/// meaning of the addr parameter and the object ownership polic
y.

   virtual Int_t           SetBranchAddress(const char *bname,vo
id *add, TBranch **ptr, TClass *realClass, EDataType datatype, B
ool_t isptr);
/// Verify the validity of the type of addr before calling SetBr
anchAddress.
/// See TTree::CheckBranchAddressType for the semantic of the re
turn value.
/// Note: See the comments in TBranchElement::SetAddress() for t
he
/// meaning of the addr parameter and the object ownership polic
y.

   template <class T> Int_t SetBranchAddress(const char *bname,
T **add, TBranch **ptr = 0) {
      TClass *cl = TClass::GetClass(typeid(T));
      EDataType type = kOther_t;
      if (cl==0) type = TDataType::GetType(typeid(T));
      return SetBranchAddress(bname,add,ptr,cl,type,true);
   }
#ifndef R__NO_CLASS_TEMPLATE_SPECIALIZATION
   // This can only be used when the template overload resolutio
```

```
n can distringuish between
   // T* and T**
   template <class T> Int_t SetBranchAddress(const char *bname,
T *add, TBranch **ptr = 0) {
      TClass *cl = TClass::GetClass(typeid(T));
      EDataType type = kOther_t;
      if (cl==0) type = TDataType::GetType(typeid(T));
      return SetBranchAddress(bname,add,ptr,cl,type,false);
   }
#endif
   virtual void            SetBranchStatus(const char* bname, Bo
ol_t status = 1, UInt_t* found = 0);
/// Set branch status to Process or DoNotProcess.
/// When reading a Tree, by default, all branches are read.
/// One can speed up considerably the analysis phase by activati
ng
/// only the branches that hold variables involved in a query.
/// bname is the name of a branch.
/// - if bname="*", apply to all branches.
/// - if bname="xxx*", apply to all branches with name starting
with xxx
/// see TRegexp for wildcarding options
/// - status = 1  branch will be processed
/// - = 0  branch will not be processed
/// __WARNING! WARNING! WARNING!__
/// SetBranchStatus is matching the branch based on match of the
 branch
/// 'name' and not on the branch hierarchy! In order to be able
to
/// selectively enable a top level object that is 'split' you ne
ed to make
/// sure the name of the top level branch is prefixed to the sub
-branches'
/// name (by adding a dot ('.') at the end of the Branch creatio
n and use the
/// corresponding bname.
/// If found is not 0, the number of branch(es) found matching t
he regular
/// expression is returned in *found AND the error message 'unkn
own branch'
```

```
/// is suppressed.

   static  void            SetBranchStyle(Int_t style = 1);  //s
tyle=0 for old branch, =1 for new branch style
   virtual Int_t           SetCacheSize(Long64_t cachesize = -1)
;
/// Set maximum size of the file cache .
/// - if cachesize = 0 the existing cache (if any) is deleted.
/// - if cachesize = -1 (default) it is set to the AutoFlush val
ue when writing
///    the Tree (default is 30 MBytes).
/// Returns:
/// - 0 size set, cache was created if possible
/// - -1 on error

   virtual Int_t           SetCacheEntryRange(Long64_t first, Lo
ng64_t last);
///interface to TTreeCache to set the cache entry range
/// Returns:
/// - 0 entry range set
/// - -1 on error

   virtual void            SetCacheLearnEntries(Int_t n=10);///
Interface to TTreeCache to set the number of entries for the lea
rning phase
   virtual void            SetChainOffset(Long64_t offset = 0) {
 fChainOffset=offset; }
   virtual void            SetCircular(Long64_t maxEntries);
/// Enable/Disable circularity for this tree.
/// if maxEntries > 0 a maximum of maxEntries is kept in one buf
fer/basket
/// per branch in memory.
///   Note that when this function is called (maxEntries>0) the
Tree
///   must be empty or having only one basket per branch.
/// if maxEntries <= 0 the tree circularity is disabled.
/// #### NOTE 1:
///  Circular Trees are interesting in online real time environm
ents
///  to store the results of the last maxEntries events.
```

```
/// #### NOTE 2:
///  Calling SetCircular with maxEntries <= 0 is necessary before

///  merging circular Trees that have been saved on files.
/// #### NOTE 3:
///  SetCircular with maxEntries <= 0 is automatically called
///  by TChain::Merge
/// #### NOTE 4:
///  A circular Tree can still be saved in a file. When read bac
k,
///  it is still a circular Tree and can be filled again.

   virtual void            SetDebug(Int_t level = 1, Long64_t mi
n = 0, Long64_t max = 9999999); // *MENU*
/// Set the debug level and the debug range.
/// For entries in the debug range, the functions TBranchElement
::Fill
/// and TBranchElement::GetEntry will print the number of bytes
filled
/// or read for each branch.

   virtual void            SetDefaultEntryOffsetLen(Int_t newdef
ault, Bool_t updateExisting = kFALSE);
/// Update the default value for the branch's fEntryOffsetLen.
/// If updateExisting is true, also update all the existing bran
ches.
/// If newdefault is less than 10, the new default value will be
 10.

   virtual void            SetDirectory(TDirectory* dir);
/// Change the tree's directory.
/// Remove reference to this tree from current directory and
/// add reference to new directory dir.  The dir parameter can
/// be 0 in which case the tree does not belong to any directory.


   virtual Long64_t        SetEntries(Long64_t n = -1);
/// Change number of entries in the tree.
/// If n >= 0, set number of entries in the tree = n.
/// If n < 0, set number of entries in the tree to match the
```

```
/// number of entries in each branch. (default for n is -1)
/// This function should be called only when one fills each bran
ch
/// independently via TBranch::Fill without calling TTree::Fill.
/// Calling TTree::SetEntries() make sense only if the number of
 entries
/// in each branch is identical, a warning is issued otherwise.
/// The function returns the number of entries.

   virtual void            SetEstimate(Long64_t nentries = 10000
00);
/// Set number of entries to estimate variable limits.
/// If n is -1, the estimate is set to be the current maximum
/// for the tree (i.e. GetEntries() + 1)
/// If n is less than -1, the behavior is undefined.

   virtual void            SetFileNumber(Int_t number = 0);
/// Set fFileNumber to number.
/// fFileNumber is used by TTree::Fill to set the file name
/// for a new file to be created when the current file exceeds f
gTreeMaxSize.
///    (see TTree::ChangeFile)
/// if fFileNumber=10, the new file name will have a suffix "_11
",
/// ie, fFileNumber is incremented before setting the file name

   virtual void            SetEventList(TEventList* list);
/// This function transfroms the given TEventList into a TEntryL
ist
/// The new TEntryList is owned by the TTree and gets deleted wh
en the tree
/// is deleted. This TEntryList can be returned by GetEntryList(
) function.

   virtual void            SetEntryList(TEntryList* list, Option
_t *opt="");/// Set an EntryList
   virtual void            SetMakeClass(Int_t make);
/// Set all the branches in this TTree to be in decomposed objec
t mode
/// (also known as MakeClass mode).
```

```
   virtual void            SetMaxEntryLoop(Long64_t maxev = kMax
Entries) { fMaxEntryLoop = maxev; } // *MENU*
   static  void            SetMaxTreeSize(Long64_t maxsize = 190
0000000);
/// Set the maximum size in bytes of a Tree file (static functio
n).
/// The default size is 100000000000LL, ie 100 Gigabytes.
/// In TTree::Fill, when the file has a size > fgMaxTreeSize,
/// the function closes the current file and starts writing into
/// a new file with a name of the style "file_1.root" if the ori
ginal
/// requested file name was "file.root".

   virtual void            SetMaxVirtualSize(Long64_t size = 0)
{ fMaxVirtualSize = size; } // *MENU*
   virtual void            SetName(const char* name); // *MENU*
/// Change the name of this tree.
   virtual void            SetNotify(TObject* obj) { fNotify = o
bj; }
   virtual void            SetObject(const char* name, const char
* title);/// Change the name and title of this tree.
   virtual void            SetParallelUnzip(Bool_t opt=kTRUE, Fl
oat_t RelSize=-1);/// Enable or disable parallel unzipping of Tr
ee buffers.
   virtual void            SetPerfStats(TVirtualPerfStats* perf);
/// Set perf stats
   virtual void            SetScanField(Int_t n = 50) { fScanFie
ld = n; } // *MENU*
   virtual void            SetTimerInterval(Int_t msec = 333) {
fTimerInterval=msec; }
   virtual void            SetTreeIndex(TVirtualIndex* index);
/// The current TreeIndex is replaced by the new index.
/// Note that this function does not delete the previous index.

   virtual void            SetWeight(Double_t w = 1, Option_t* o
ption = "");
/// Set tree weight.
/// The weight is used by TTree::Draw to automatically weight ea
ch
```

```
/// selected entry in the resulting histogram.
/// This function is redefined by TChain::SetWeight. In case of a

/// TChain, an option "global" may be specified to set the same
weight
/// for all trees in the TChain instead of the default behaviour
/// using the weights of each tree in the chain (see TChain::Set
Weight).

   virtual void            SetUpdate(Int_t freq = 0) { fUpdate =
 freq; }
   virtual void            Show(Long64_t entry = -1, Int_t lenma
x = 20);
/// Print values of all active leaves for entry.
/// - if entry==-1, print current entry (default)
/// - if a leaf is an array, a maximum of lenmax elements is pri
nted.

   virtual void            StartViewer(); // *MENU*
/// Start the TTreeViewer on this tree.
/// - ww is the width of the canvas in pixels
/// - wh is the height of the canvas in pixels

   virtual Int_t           StopCacheLearningPhase();
/// Stop the cache learning phase
/// Returns:
/// - 0 learning phase stopped or not active
/// - -1 on error

   virtual Int_t           UnbinnedFit(const char* funcname, con
st char* varexp, const char* selection = "", Option_t* option =
"", Long64_t nentries = kMaxEntries, Long64_t firstentry = 0);
/// Unbinned fit of one or more variable(s) from a tree.
/// funcname is a TF1 function.
/// See TTree::Draw for explanations of the other parameters.
/// Fit the variable varexp using the function funcname using the

/// selection cuts given by selection.
/// The list of fit options is given in parameter option.
/// - option = "Q" Quiet mode (minimum printing)
```

```
/// - option = "V" Verbose mode (default is between Q and V)
/// - option = "E" Perform better Errors estimation using Minos
technique
/// - option = "M" More. Improve fit results
/// With this setup:
/// - Parameters 0->3 can vary freely
/// - Parameter 4 has boundaries [-10,-4] with initial value -8
/// - Parameter 5 is fixed to 100.
/// For the fit to be meaningful, the function must be self-norm
alized.
/// 1, 2 and 3 Dimensional fits are supported. See also TTree::F
it
/// Return status:
/// - The function return the status of the fit in the following
 form
///   fitResult = migradResult + 10*minosResult + 100*hesseResul
t + 1000*improveResult
/// - The fitResult is 0 is the fit is OK.
/// - The fitResult is negative in case of an error not connecte
d with the fit.
/// - The number of entries used in the fit can be obtained via
mytree.GetSelectedRows();
/// - If the number of selected entries is null the function ret
urns -1

   void                    UseCurrentStyle();/// Replace current
 attributes by current style.
   virtual Int_t           Write(const char *name=0, Int_t optio
n=0, Int_t bufsize=0);
/// Write this object to the current directory. For more see TOb
ject::Write
/// If option & kFlushBasket, call FlushBasket before writing th
e tree.

   virtual Int_t           Write(const char *name=0, Int_t optio
n=0, Int_t bufsize=0) const;
/// Write this object to the current directory. For more see TOb
ject::Write
/// Write calls TTree::FlushBaskets before writing the tree.
```

code

```
/// You can specify boundary limits for some or all parameters v
ia

func->SetParLimits(p_number, parmin, parmax);

/// if parmin>=parmax, the parameter is fixed

/// Note that you are not forced to fix the limits for all param
eters.
/// For example, if you fit a function with 6 parameters, you ca
n do:

func->SetParameters(0,3.1,1.e-6,0.1,-8,100);
func->SetParLimits(4,-10,-4);
func->SetParLimits(5, 1,1);

/// i.e. It must have the same integral regardless of the parame
ter
/// settings.  Otherwise the fit will effectively just maximize
the
/// area.

/// It is mandatory to have a normalization variable
/// which is fixed for the fit.  e.g.

TF1* f1 = new TF1("f1", "gaus(0)/sqrt(2*3.14159)/[2]", 0, 5);
f1->SetParameters(1, 3.1, 0.01);
f1->SetParLimits(0, 1, 1); // fix the normalization parameter to
 1
data->UnbinnedFit("f1", "jpsimass", "jpsipt>3.0");
```

```
// This gives the possibility to play with more than one index,
e.g.,

TVirtualIndex* oldIndex = tree.GetTreeIndex();
tree.SetTreeIndex(newIndex);
tree.Draw();
tree.SetTreeIndex(oldIndex);
tree.Draw(); etc
```

```
/// Assume a tree T with sub-branches a,b,c,d,e,f,g,etc..
/// when doing T.GetEntry(i) all branches are read for entry i.
/// to read only the branches c and e, one can do

T.SetBranchStatus("*",0); //disable all branches
T.SetBranchStatus("c",1);
T.setBranchStatus("e",1);
T.GetEntry(i);

/// bname is interpreted as a wildcarded TRegexp (see TRegexp::M
akeWildcard).
/// Thus, "a*b" or "a.*b" matches branches starting with "a" and
 ending with
/// "b", but not any other branch with an "a" followed at some p
oint by a
/// "b". For this second behavior, use "*a*b*". Note that TRegEx
p does not
/// support '|', and so you cannot select, e.g. track and shower
 branches
/// with "track|shower".

/// I.e If your Tree has been created in split mode with a paren
t branch "parent."
/// (note the trailing dot).

T.SetBranchStatus("parent",1);

/// will not activate the sub-branches of "parent". You should d
o:
```

```
T.SetBranchStatus("parent*",1);

/// Without the trailing dot in the branch creation you have no
choice but to
/// call SetBranchStatus explicitly for each of the sub branches.


/// An alternative to this function is to read directly and only
/// the interesting branches. Example:

TBranch *brc = T.GetBranch("c");
TBranch *bre = T.GetBranch("e");
brc->GetEntry(i);
bre->GetEntry(i);
```

```
tree->SetAlias("x1","(tdc1[1]-tdc1[0])/49");
tree->SetAlias("y1","(tdc1[3]-tdc1[2])/47");
tree->SetAlias("x2","(tdc2[1]-tdc2[0])/49");
tree->SetAlias("y2","(tdc2[3]-tdc2[2])/47");
tree->Draw("y2-y1:x2-x1");

tree->SetAlias("theGoodTrack","event.fTracks[3]");
tree->Draw("theGoodTrack.fPx"); // same as "event.fTracks[3].fPx"
```

```
/// To fill a TTree with multiple input text files, proceed as i
ndicated above
/// for the first input file and omit the second argument for su
bsequent calls

T.ReadFile("file1.dat","branch descriptor");
T.ReadFile("file2.dat");
```

```
/// ## NOTE1
/// It may be more interesting to invoke directly the other Proc
ess function
/// accepting a TSelector* as argument.eg

MySelector *selector = (MySelector*)TSelector::GetSelector(filen
ame);
selector->CallSomeFunction(..);
mytree.Process(selector,..);

/// ## NOTE2
/// One should not call this function twice with the same select
or file
/// in the same script. If this is required, proceed as indicate
d in NOTE1,
/// by getting a pointer to the corresponding TSelector,eg

void stubs1() {
   TSelector *selector = TSelector::GetSelector("h1test.C");
   TFile *f1 = new TFile("stubs_nood_le1.root");
   TTree *h1 = (TTree*)f1->Get("h1");
   h1->Process(selector);
   TFile *f2 = new TFile("stubs_nood_le1_coarse.root");
    TTree *h2 = (TTree*)f2->Get("h1");
    h2->Process(selector);
}

/// or use ACLIC to compile the selector

void stubs2() {
    TFile *f1 = new TFile("stubs_nood_le1.root");
    TTree *h1 = (TTree*)f1->Get("h1");
    h1->Process("h1test.C+");
   TFile *f2 = new TFile("stubs_nood_le1_coarse.root");
   TTree *h2 = (TTree*)f2->Get("h1");
   h2->Process("h1test.C+");
}
```

```
/// you can retrieve a pointer to the created object via:

TPrincipal *principal = (TPrincipal*)gROOT->GetListOfSpecials()-
>FindObject("principal");
```

```
/// For example with Event.root, if

Double_t somePx = fTracks.fPx[2];

/// is executed by one of the method of the skeleton,
/// somePx will updated with the current value of fPx of the 3rd
 track.

/// Both macrofilename and the optional cutfilename are expected
 to be
/// the name of source files which contain at least a free stand
ing
/// function with the signature:

x_t macrofilename(); // i.e function with the same name as the f
ile

/// and

y_t cutfilename();   // i.e function with the same name as the f
ile

/// x_t and y_t needs to be types that can convert respectively
to a double
/// and a bool (because the skeleton uses:

if (cutfilename()) htemp->Fill(macrofilename());

/// These two functions are run in a context such that the branc
h names are
/// available as local variables of the correct (read-only) type.
```

```
/// Note that if you use the same 'variable' twice, it is more e
fficient
/// to 'cache' the value. For example:

Int_t n = fEventNumber; // Read fEventNumber
if (n<10 || n>10) { ... }

/// is more efficient than

if (fEventNumber<10 || fEventNumber>10)

/// Also, optionally, the generated selector will also call meth
ods named
/// macrofilename_methodname in each of 6 main selector methods
if the method
/// macrofilename_methodname exist (Where macrofilename is strip
ped of its
/// extension).

/// To draw px using the file hsimple.root (generated by the
/// hsimple.C tutorial), we need a file named hsimple.cxx:

double hsimple() {
   return px;
}

/// MakeProxy can then be used indirectly via the TTree::Draw in
terface
/// as follow:

new TFile("hsimple.root")
ntuple->Draw("hsimple.cxx");
```

```
/// To use this function:
/// - Open your tree file (eg: TFile f("myfile.root");)
/// - T->MakeClass("MyClass");

/// where T is the name of the TTree in file myfile.root,
/// and MyClass.h, MyClass.C the name of the files created by th
is function.
/// In a ROOT session, you can do:

root > .L MyClass.C
root > MyClass* t = new MyClass;
root > t->GetEntry(12); // Fill data members of t with entry num
ber 12.
root > t->Show();       // Show values of entry 12.
root > t->Show(16);     // Read and show values of entry 16.
root > t->Loop();       // Loop on all entries.
```

```
/// To activate/deactivate one or more branches, use TBranch::Se
tBranchStatus
/// For example, if you have a Tree with several hundred branche
s, and you
/// are interested only by branches named "a" and "b", do

mytree.SetBranchStatus("*",0); //disable all branches
mytree.SetBranchStatus("a",1);
mytree.SetBranchStatus("b",1);

/// when calling mytree.GetEntry(i); only branches "a" and "b" w
ill be read.

/// __WARNING!!__
/// If your Tree has been created in split mode with a parent br
anch "parent.",

mytree.SetBranchStatus("parent",1);

/// will not activate the sub-branches of "parent". You should d
o:
```

```
mytree.SetBranchStatus("parent*",1);

/// Without the trailing dot in the branch creation you have no
choice but to
/// call SetBranchStatus explicitly for each of the sub branches.


/// An alternative is to call directly

brancha.GetEntry(i)
branchb.GetEntry(i);


/// Consider the example in $ROOTSYS/test/Event.h
/// The top level branch in the tree T is declared with:

Event *event = 0;  //event must be null or point to a valid obje
ct
                   //it must be initialized
T.SetBranchAddress("event",&event);

/// When reading the Tree, one can choose one of these 3 options:

///
/// ## OPTION 1

for (Long64_t i=0;i<nentries;i++) {
   T.GetEntry(i);
   // the object event has been filled at this point
}

/// The default (recommended). At the first entry an object of t
he class
/// Event will be created and pointed by event. At the following
 entries,
/// event will be overwritten by the new data. All internal memb
ers that are
/// TObject* are automatically deleted. It is important that the
se members
```

```
/// be in a valid state when GetEntry is called. Pointers must b
e correctly
/// initialized. However these internal members will not be dele
ted if the
/// characters "->" are specified as the first characters in the
 comment
/// field of the data member declaration.
///
/// If "->" is specified, the pointer member is read via pointer
->Streamer(buf).
/// In this case, it is assumed that the pointer is never null (
case of
/// pointer TClonesArray *fTracks in the Event example). If "->"
 is not
/// specified, the pointer member is read via buf >> pointer. In
 this case
/// the pointer may be null. Note that the option with "->" is f
aster to
/// read or write and it also consumes less space in the file.
///
/// ## OPTION 2
///
/// The option AutoDelete is set

TBranch *branch = T.GetBranch("event");
branch->SetAddress(&event);
branch->SetAutoDelete(kTRUE);
for (Long64_t i=0;i<nentries;i++) {
   T.GetEntry(i);
   // the object event has been filled at this point
}

/// In this case, at each iteration, the object event is deleted
 by GetEntry
/// and a new instance of Event is created and filled.

/// ## OPTION 3

/// Same as option 1, but you delete yourself the event.
```

```
for (Long64_t i=0;i<nentries;i++) {
   delete event;
   event = 0;  // EXTREMELY IMPORTANT
   T.GetEntry(i);
    // the object event has been filled at this point
}

/// It is strongly recommended to use the default option 1. It h
as the
/// additional advantage that functions like TTree::Draw (intern
ally calling
/// TTree::GetEntry) will be functional even when the classes in
 the file are
/// not available.

/// Note: See the comments in TBranchElement::SetAddress() for t
he
/// object ownership policy of the underlying (user) data.
```

```
TTree::TClusterIterator clusterIter = tree->GetClusterIterator(e
ntry);
Long64_t clusterStart;
while( (clusterStart = clusterIter()) < tree->GetEntries() ) {
   printf("The cluster starts at %lld and ends at %lld (inclusiv
e)\n",clusterStart,clusterIter.GetNextEntry()-1);
}
```

```
tree.Fit(pol4,sqrt(x)>>hsqrt,y>0)

/// will fit sqrt(x) and save the histogram as "hsqrt" in the cu
rrent
/// directory.

/// See also TTree::UnbinnedFit
```

example

TTreePlayer

TVector2

TVector3

TVectorT

TVirtualFitter

README

```
// System predefined widget message types. Message types are con
stants
// that indicate which widget sent the message and by which widg
et
// function (sub-message). Make sure your own message types don'
t clash
// whith the ones defined in this file. ROOT reserves all messag
e ids
// between 0 - 1000. User defined messages should be in the rang
e
// 1001 - 10000. Sub-messages must always be in the range 1-255.

// To use MK_MSG() just cast your message id's to an EWidgetMess
ageType.

// WidgetMessageTypes
enum EWidgetMessageTypes {
   kC_COMMAND          = 1,
      kCM_MENU            = 1,
      kCM_MENUSELECT      = 2,
      kCM_BUTTON          = 3,
      kCM_CHECKBUTTON     = 4,
      kCM_RADIOBUTTON     = 5,
      kCM_LISTBOX         = 6,
      kCM_COMBOBOX        = 7,
      kCM_TAB             = 8,
   kC_HSCROLL          = 2,
   kC_VSCROLL          = 3,
      kSB_LINEUP          = 1,
      kSB_LINEDOWN        = 2,
      kSB_PAGEUP          = 3,
      kSB_PAGEDOWN        = 4,
      kSB_SLIDERTRACK     = 5,
      kSB_SLIDERPOS       = 6,
   kC_TEXTENTRY        = 4,
```

```
        kTE_TEXTCHANGED      = 1,
        kTE_ENTER            = 2,
        kTE_TAB              = 3,
        kTE_KEY              = 4,
    kC_CONTAINER         = 5,
        kCT_ITEMCLICK        = 1,
        kCT_ITEMDBLCLICK     = 2,
        kCT_SELCHANGED       = 3,
        kCT_KEY              = 4,
    kC_HSLIDER           = 6,
    kC_VSLIDER           = 7,
        kSL_POS              = 1,
        kSL_TRACK            = 2,
        kSL_PRESS            = 3,
        kSL_RELEASE          = 4,
        kSL_POINTER          = 5,
    kC_LISTTREE          = 8,
    kC_TEXTVIEW          = 9,
        kTXT_ISMARKED        = 1,
        kTXT_DATACHANGE      = 2,
        kTXT_CLICK2          = 3,
        kTXT_CLICK3          = 4,
        kTXT_F3              = 5,
        kTXT_OPEN            = 6,
        kTXT_CLOSE           = 7,
        kTXT_SAVE            = 8,
    kC_COLORSEL          = 10,
        kCOL_CLICK           = 1,
        kCOL_SELCHANGED      = 2,
    kC_PATTERNSEL        = 11,
        kPAT_CLICK           = 1,
        kPAT_SELCHANGED      = 2,
    kC_MARKERSEL         = 12,
        kMAR_CLICK           = 1,
        kMAR_SELCHANGED      = 2,
    kC_POPUP             = 13,
        kPOP_HIDE            = 1,
    kC_DOCK              = 14,
        kDOCK_DOCK           = 1,
        kDOCK_UNDOCK         = 2,
```

```
        kDOCK_SHOW          = 3,
        kDOCK_HIDE          = 4,
    kC_MDI              = 15,
        kMDI_CURRENT        = 1,
        kMDI_CREATE         = 2,
        kMDI_CLOSE          = 4,
        kMDI_RESTORE        = 8,
        kMDI_MOVE           = 16,
        kMDI_SIZE           = 32,
        kMDI_MINIMIZE       = 64,
        kMDI_MAXIMIZE       = 128,
        kMDI_HELP           = 256,
        kMDI_MENU           = 512,
    kC_USER             = 1001,
    kC_MSGMAX           = 10000
};
```

- TGCheckButton
- TGComboBox
- TGGroupFrame
- TGIcon
- TGLabel
- TGListBox
- TGMenuBar
- TGNumberEntry
- TGPicture
- TGPicturePool
- TGPopupMenu
- TGRadioButton
- TGSplitButton
- TGTextEntry
- TGTripleSlider
- TGVButtonGroup
- TGTextButton

```
//////////////////////////////////////////////////////////////////
//////////
```

```
//
        //
// TGButton, TGTextButton, TGPictureButton, TGCheckButton TGRadi
oButton //
// and TGSplitButton
        //
//
        //
// This header defines all GUI button widgets.
        //
//
        //
// TGButton is a button abstract base class. It defines general
button  //
// behaviour.
        //
//
        //
// Selecting a text or picture button will generate the event:
        //
// kC_COMMAND, kCM_BUTTON, button id, user data.
        //
//
        //
// Selecting a check button will generate the event:
        //
// kC_COMMAND, kCM_CHECKBUTTON, button id, user data.
        //
//
        //
// Selecting a radio button will generate the event:
        //
// kC_COMMAND, kCM_RADIOBUTTON, button id, user data.
        //
//
        //
// If a command string has been specified (via SetCommand()) the
n this  //
// command string will be executed via the interpreter whenever
a       //
```

```
// button is selected. A command string can contain the macros:
        //
// $MSG   -- kC_COMMAND, kCM[CHECK|RADIO]BUTTON packed message
        //
//           (use GET_MSG() and GET_SUBMSG() to unpack)
        //
// $PARM1 -- button id
        //
// $PARM2 -- user data pointer
        //
// Before executing these macros are expanded into the respectiv
e       //
// Long_t's
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGColorPalette, TGColorPick and TGColorDialog.
        //
//
        //
// The TGColorPalette is a widget showing an matrix of color cel
ls. The //
// colors can be set and selected.
        //
//
        //
// The TGColorPick is a widget which allows a color to be picked
 from   //
// HLS space. It consists of two elements: a color map window fr
om      //
// where the user can select the hue and saturation level of a c
olor,   //
// and a slider to select color's lightness.
```

```
        //
//
        //
// Selecting a color in these two widgets will generate the even
t:      //
// kC_COLORSEL, kCOL_CLICK, widget id, 0.
        //
// and the signal:
        //
// ColorSelected(Pixel_t color)
        //
//
        //
// The TGColorDialog presents a full featured color selection di
alog.   //
// It uses 2 TGColorPalette's and the TGColorPick widgets.
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGColorFrame, TG16ColorSelector, TGColorPopup and TGColorSele
ct.     //
//
        //
// The TGColorFrame is a small frame with border showing a speci
fic     //
// color.
        //
//
        //
// The TG16ColorSelector is a composite frame with 16 TGColorFra
mes.    //
//
        //
```

```
// The TGColorPopup is a popup containing a TG16ColorSelector an
d a     //
// "More..." button which popups up a TGColorDialog allowing cus
tom     //
// color selection.
        //
//
        //
// The TGColorSelect widget is like a checkbutton but instead of
 the    //
// check mark there is color area with a little down arrow. When
        //
// clicked on the arrow the TGColorPopup pops up.
        //
//
        //
// Selecting a color in this widget will generate the event:
        //
// kC_COLORSEL, kCOL_SELCHANGED, widget id, pixel.
        //
// and the signal:
        //
// ColorSelected(Pixel_t pixel)
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGComboBox, TGComboBoxPopup
        //
//
        //
// A combobox (also known as a drop down listbox) allows the sel
ection  //
// of one item out of a list of items. The selected item is visi
```

```
ble in  //
// a little window. To view the list of possible items one has t
o click //
// on a button on the right of the little window. This will drop
 down   //
// a listbox. After selecting an item from the listbox the box w
ill     //
// disappear and the newly selected item will be shown in the li
ttle    //
// window.
        //
//
        //
// The TGComboBox is user callable. The TGComboBoxPopup is a ser
vice    //
// class of the combobox.
        //
//
        //
// Selecting an item in the combobox will generate the event:
        //
// kC_COMMAND, kCM_COMBOBOX, combobox id, item id.
        //
//
        //
//////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////
//////////
//
        //
// TGDoubleSlider, TGDoubleVSlider and TGDoubleHSlider
        //
//
        //
// DoubleSlider widgets allow easy selection of a min and a max
value   //
// out of a range.
        //
```

```
// DoubleSliders can be either horizontal or vertical oriented a
nd      //
// there is a choice of three different types of tick marks.
        //
//
        //
// To change the min value press the mouse near to the left / bo
ttom    //
// edge of the slider.
        //
// To change the max value press the mouse near to the right / t
op      //
// edge of the slider.
        //
// To change both values simultaneously press the mouse near to
the     //
// center of the slider.
        //
//
        //
// TGDoubleSlider is an abstract base class. Use the concrete
        //
// TGDoubleVSlider and TGDoubleHSlider.
        //
//
        //
// Dragging the slider will generate the event:
        //
// kC_VSLIDER, kSL_POS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_POS, slider id, 0  (for horizontal slider)
        //
//
        //
// Pressing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)
        //
```

```
//
        //
// Releasing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider
)       //
//
        //
// Use the functions GetMinPosition(), GetMaxPosition() and
        //
// GetPosition() to retrieve the position of the slider.
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGListBox, TGLBContainer, TGLBEntry and TGTextLBEntry
        //
//
        //
// A listbox is a box, possibly with scrollbar, containing entri
es.     //
// Currently entries are simple text strings (TGTextLBEntry).
        //
// A TGListBox looks a lot like a TGCanvas. It has a TGViewPort
        //
// containing a TGLBContainer which contains the entries and it
also    //
// has a vertical scrollbar which becomes visible if there are m
ore     //
// items than fit in the visible part of the container.
        //
//
```

```
        //
// The TGListBox is user callable. The other classes are service
        //
// classes of the listbox.
        //
//
        //
// Selecting an item in the listbox will generate the event:
        //
// kC_COMMAND, kCM_LISTBOX, listbox id, item id.
        //
//
        //
//////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////
//////////
//
        //
// TGListTree and TGListTreeItem
        //
//
        //
// A list tree is a widget that can contain a number of items
        //
// arranged in a tree structure. The items are represented by sm
all     //
// folder icons that can be either open or closed.
        //
//
        //
// The TGListTree is user callable. The TGListTreeItem is a serv
ice     //
// class of the list tree.
        //
//
        //
// A list tree can generate the following events:
        //
```

```
// kC_LISTTREE, kCT_ITEMCLICK, which button, location (y<<16|x).
        //
// kC_LISTTREE, kCT_ITEMDBLCLICK, which button, location (y<<16|
x).     //
//
        //
//////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////
//////////
//
        //
// TGListView, TGLVContainer and TGLVEntry
        //
//
        //
// A list view is a widget that can contain a number of items
        //
// arranged in a grid or list. The items can be represented eith
er      //
// by a string or by an icon.
        //
//
        //
// The TGListView is user callable. The other classes are servic
e       //
// classes of the list view.
        //
//
        //
// A list view can generate the following events:
        //
// kC_CONTAINER, kCT_SELCHANGED, total items, selected items.
        //
// kC_CONTAINER, kCT_ITEMCLICK, which button, location (y<<16|x)
.       //
// kC_CONTAINER, kCT_ITEMDBLCLICK, which button, location (y<<16
|x).    //
//
```

```
        //
////////////////////////////////////////////////////////////////
//////////

////////////////////////////////////////////////////////////////
//////////
//
        //
// TGMenuBar, TGPopupMenu, TGMenuTitle and TGMenuEntry
        //
//
        //
// This header contains all different menu classes.
        //
//
        //
// Selecting a menu item will generate the event:
        //
// kC_COMMAND, kCM_MENU, menu id, user data.
        //
//
        //
////////////////////////////////////////////////////////////////
//////////

////////////////////////////////////////////////////////////////
//////////
//
        //
// TGScrollBar and TGScrollBarElement
        //
//
        //
// The classes in this file implement scrollbars. Scrollbars can
 be     //
// either placed horizontal or vertical. A scrollbar contains th
ree     //
// TGScrollBarElements: The "head", "tail" and "slider". The hea
d and   //
// tail are fixed at either end and have the typical arrows in t
```

```
hem.    //
//
        //
// The TGHScrollBar will generate the following event messages:
        //
// kC_HSCROLL, kSB_SLIDERPOS, position, 0
        //
// kC_HSCROLL, kSB_SLIDERTRACK, position, 0
        //
//
        //
// The TGVScrollBar will generate the following event messages:
        //
// kC_VSCROLL, kSB_SLIDERPOS, position, 0
        //
// kC_VSCROLL, kSB_SLIDERTRACK, position, 0
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGSlider, TGVSlider and TGHSlider
        //
//
        //
// Slider widgets allow easy selection of a range.
        //
// Sliders can be either horizontal or vertical oriented and the
re is   //
// a choice of two different slider types and three different ty
pes     //
// of tick marks.
        //
//
        //
```

```
// TGSlider is an abstract base class. Use the concrete TGVSlide
r and   //
// TGHSlider.
        //
//
        //
// Dragging the slider will generate the event:
        //
// kC_VSLIDER, kSL_POS, slider id, position  (for vertical slide
r)      //
// kC_HSLIDER, kSL_POS, slider id, position  (for horizontal sli
der)    //
//
        //
// Pressing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)
        //
//
        //
// Releasing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider
)       //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGTab, TGTabElement, TGTabLayout
        //
//
```

```
        //
// A tab widget contains a set of composite frames each with a l
ittle   //
// tab with a name (like a set of folders with tabs).
        //
//
        //
// The TGTab is user callable. The TGTabElement and TGTabLayout
are     //
// is a service classes of the tab widget.
        //
//
        //
// Clicking on a tab will bring the associated composite frame t
o the   //
// front and generate the following event:
        //
// kC_COMMAND, kCM_TAB, tab id, 0.
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGTextEntry
        //
//
        //
// A TGTextEntry is a one line text input widget.
        //
//
        //
// Changing text in the text entry widget will generate the even
t:      //
// kC_TEXTENTRY, kTE_TEXTCHANGED, widget id, 0.
        //
```

```
// Hitting the enter key will generate:
        //
// kC_TEXTENTRY, kTE_ENTER, widget id, 0.
        //
// Hitting the tab key will generate:
        //
// kC_TEXTENTRY, kTE_TAB, widget id, 0.
        //
//
        //
//////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////
//////////
//
        //
// TGTripleVSlider and TGTripleHSlider
        //
//
        //
// TripleSlider inherit from DoubleSlider widgets and allow easy
        //
// selection of a min, max and pointer value out of a range.
        //
// The pointer position can be constrained to edges of slider an
d / or  //
// can be relative to the slider position.
        //
//
        //
// To change the min value press the mouse near to the left / bo
ttom    //
// edge of the slider.
        //
// To change the max value press the mouse near to the right / t
op      //
// edge of the slider.
        //
// To change both values simultaneously press the mouse near to
```

```
the     //
// center of the slider.
        //
// To change pointer value press the mouse on the pointer and dr
ag it   //
// to the desired position
        //
//
        //
// Dragging the slider will generate the event:
        //
// kC_VSLIDER, kSL_POS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_POS, slider id, 0  (for horizontal slider)
        //
//
        //
// Pressing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)
        //
//
        //
// Releasing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider
)       //
//
        //
// Moving the pointer will generate the event:
        //
// kC_VSLIDER, kSL_POINTER, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_POINTER, slider id, 0  (for horizontal slider
)       //
//
```

```
        //
// Use the functions GetMinPosition(), GetMaxPosition() and
        //
// GetPosition() to retrieve the position of the slider.
        //
// Use the function GetPointerPosition() to retrieve the positio
n of    //
// the pointer
        //
//
        //
//////////////////////////////////////////////////////////////
//////////

//////////////////////////////////////////////////////////////
//////////
//
        //
// TGView
        //
//
        //
// A TGView provides the infrastructure for text viewer and edit
or      //
// widgets. It provides a canvas (TGViewFrame) and (optionally)
a       //
// vertical and horizontal scrollbar and methods for marking and
        //
// scrolling.
        //
//
        //
// The TGView (and derivatives) will generate the following
        //
// event messages:
        //
// kC_TEXTVIEW, kTXT_ISMARKED, widget id, [true|false]
        //
// kC_TEXTVIEW, kTXT_DATACHANGE, widget id, 0
        //
```

```
// kC_TEXTVIEW, kTXT_CLICK2, widget id, position (y << 16) | x)
        //
// kC_TEXTVIEW, kTXT_CLICK3, widget id, position (y << 16) | x)
        //
// kC_TEXTVIEW, kTXT_F3, widget id, true
        //
//
        //
//////////////////////////////////////////////////////////////
//////////
```

```
static const char *gFonts[][2] = {    //   unix name,     name
   { "",                                                  ""
              }, //not used
   { "-*-times-medium-i-*-*-12-*-*-*-*-*-*-*",     "1. times ita
lic"          },
   { "-*-times-bold-r-*-*-12-*-*-*-*-*-*-*",       "2. times bol
d"            },
   { "-*-times-bold-i-*-*-12-*-*-*-*-*-*-*",       "3. times bol
d italic"     },
   { "-*-helvetica-medium-r-*-*-12-*-*-*-*-*-*-*", "4. helvetica"
             },
   { "-*-helvetica-medium-o-*-*-12-*-*-*-*-*-*-*", "5. helvetica
 italic"      },
   { "-*-helvetica-bold-r-*-*-12-*-*-*-*-*-*-*",   "6. helvetica
 bold"        },
   { "-*-helvetica-bold-o-*-*-12-*-*-*-*-*-*-*",   "7. helvetica
 bold italic" },
   { "-*-courier-medium-r-*-*-12-*-*-*-*-*-*-*",   "8. courier"
              },
   { "-*-courier-medium-o-*-*-12-*-*-*-*-*-*-*",   "9. courier i
talic"        },
   { "-*-courier-bold-r-*-*-12-*-*-*-*-*-*-*",     "10. courier
bold"         },
   { "-*-courier-bold-o-*-*-12-*-*-*-*-*-*-*",     "11. courier
bold italic"  },
   { "-*-symbol-medium-r-*-*-12-*-*-*-*-*-*-*",    "12. symbol"
              },
   { "-*-times-medium-r-*-*-12-*-*-*-*-*-*-*",     "13. times"
              },
   { 0, 0}
};
```

```
TGGC *fTextGC;
const TGFont *font = gClient->GetFont("-*-times-bold-r-*-*-18-*-
*-*-*-*-*-*");
if (!font) font = gClient->GetResourcePool()->GetDefaultFont();
FontStruct_t labelfont = font->GetFontStruct();
GCValues_t   gval;
gval.fMask = kGCBackground | kGCFont | kGCForeground;
gval.fFont = font->GetFontHandle();
gClient->GetColorByName("yellow", gval.fBackground);
fTextGC = gClient->GetGC(&gval, kTRUE);
ULong_t bcolor, ycolor;
gClient->GetColorByName("yellow", ycolor);
gClient->GetColorByName("blue", bcolor);
fStatus = new TGLabel(frame, "OwnFont & Bck/ForgrColor", fTextGC
->GetGC(), labelfont, kChildFrame, bcolor);//
fStatus->SetTextColor(ycolor);
// fStatus->ChangeOptions(fStatus->GetOptions() | kFixedSize);
// fStatus->Resize(350, 80);
frame->AddFrame(fStatus, new TGLayoutHints(kLHintsNormal, 5, 5, 3
, 4));
```

TGButton

```
//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGButton, TGTextButton, TGPictureButton, TGCheckButton,
        //
// TGRadioButton and TGSplitButton
        //
//
        //
// This header defines all GUI button widgets.
        //
//
        //
// TGButton is a button abstract base class. It defines general
button  //
// behaviour.
        //
//
        //
// TGTextButton and TGPictureButton yield an action as soon as t
hey are //
// clicked. These buttons usually provide fast access to frequen
tly     //
// used or critical commands. They may appear alone or placed in
 a      //
// group.
        //
//
        //
// The action they perform can be inscribed with a meaningful to
oltip   //
// set by SetToolTipText(const char* text, Long_t delayms=400).
        //
//
```

```
        //
// The text button has a label indicating the action to be taken
 when   //
// the button is pressed. The text can be a hot string ("&Exit")
 that   //
// defines the label "Exit" and keyboard mnemonics Alt+E for but
ton     //
// selection. A button label can be changed by SetText(new_label
).      //
//
        //
// Selecting a text or picture button will generate the event:
        //
// kC_COMMAND, kCM_BUTTON, button id, user data.
        //
//
        //
// The purpose of TGCheckButton and TGRadioButton is for selecti
ng      //
// different options. Like text buttons, they have text or hot s
tring   //
// as a label.
        //
//
        //
// Radio buttons are grouped usually in logical sets of two or m
ore     //
// buttons to present mutually exclusive choices.
        //
//
        //
// Selecting a check button will generate the event:
        //
// kC_COMMAND, kCM_CHECKBUTTON, button id, user data.
        //
//
        //
// Selecting a radio button will generate the event:
        //
// kC_COMMAND, kCM_RADIOBUTTON, button id, user data.
```

```
        //
//
        //
// If a command string has been specified (via SetCommand()) the
n this  //
// command string will be executed via the interpreter whenever
a       //
// button is selected. A command string can contain the macros:
        //
// $MSG   -- kC_COMMAND, kCM[CHECK|RADIO]BUTTON packed message
        //
//           (use GET_MSG() and GET_SUBMSG() to unpack)
        //
// $PARM1 -- button id
        //
// $PARM2 -- user data pointer
        //
// Before executing these macros are expanded into the respectiv
e       //
// Long_t's
        //
//
        //
// TGSplitButton implements a button with added menu functionali
ty.     //
// There are 2 modes of operation available.
        //
//
        //
// If the button is split, a menu will popup when the menu area
of the  //
// button is clicked. Activating a menu item changes the functio
nality  //
// of the button by having it emit a additional signal when it i
s       //
// clicked. The signal emitted when the button is clicked, is th
e       //
// ItemClicked(Int_t) signal with a different fixed value for th
e       //
// Int_t that corresponds to the id of the activated menu entry.
```

```
        //
//
        //
// If the button is not split, clicking it will popup the menu a
nd the  //
// ItemClicked(Int_t) signal will be emitted when a menu entry i
s       //
// acitvated. The value of the Int_t is again equal to the value
 of     //
// the id of the activated menu entry.
        //
//
        //
// The mode of operation of a SplitButton can be changed on the
fly     //
// by calling the SetSplit(Bool_t) method.
        //
//////////////////////////////////////////////////////////////
//////////
```

TGButton 继承 TGFrame, TGWidget ， friend TGButtonGroup

TGTextButton 继承 TGButton

TGPictureButton 继承 TGButton

TGCheckButton 继承 TGButton

TGRadioButton 继承 TGButton

TGSplitButton 继承 TGTextButton

class

```
//--- Button states

enum EButtonState {
   kButtonUp,
   kButtonDown,
   kButtonEngaged,
   kButtonDisabled
};
```

TGButton

```
   static const TGGC   &GetDefaultGC();/// Return default graphi
cs context.
   static const TGGC   &GetHibckgndGC();/// Return graphics cont
ext for highlighted frame background.

   TGButton(const TGWindow *p = 0, Int_t id = -1, GContext_t nor
m = GetDefaultGC()(),
            UInt_t option = kRaisedFrame | kDoubleBorder);
   virtual ~TGButton();

   virtual Bool_t       HandleButton(Event_t *event);/// Handle
mouse button event.
   virtual Bool_t       HandleCrossing(Event_t *event);/// Handl
e mouse crossing event.
   virtual void         SetUserData(void *userData) { fUserData
= userData; }
   virtual void        *GetUserData() const { return fUserData;
}
   virtual void         SetToolTipText(const char *text, Long_t
delayms = 400);  //*MENU*
/// Set tool tip text associated with this button. The delay is
in
/// milliseconds (minimum 250). To remove tool tip call method w
ith
/// text = 0.

   virtual TGToolTip   *GetToolTip() const { return fTip; }
   virtual void         SetState(EButtonState state, Bool_t emit
```

```
 = kFALSE);/// Set button state.
   virtual EButtonState GetState() const { return fState; }
   virtual void         AllowStayDown(Bool_t a) { fStayDown = a;
 }
   virtual void         SetGroup(TGButtonGroup *gr);/// Sets new
 button-group for this button.
   TGButtonGroup       *GetGroup() const { return fGroup; }

   virtual Bool_t       IsDown() const;// { return !(fOptions &
kRaisedFrame); }
   virtual void         SetDown(Bool_t on = kTRUE, Bool_t emit =
 kFALSE);
   virtual Bool_t       IsOn() const { return IsDown(); }
   virtual void         SetOn(Bool_t on = kTRUE,  Bool_t emit =
kFALSE) { SetDown(on, emit); }
   virtual Bool_t       IsToggleButton() const { return kFALSE;
}
   virtual Bool_t       IsExclusiveToggle() const { return kFALS
E; }
   virtual void         Toggle(Bool_t emit = kFALSE) { SetDown(I
sDown() ? kFALSE : kTRUE, emit); }
   virtual void         SetEnabled(Bool_t e = kTRUE); //*TOGGLE*
 *GETTER=IsEnabled
/// Set enabled or disabled state of button

   virtual UInt_t       GetStyle() const { return fStyle; }
   virtual void         SetStyle(UInt_t newstyle);/// Set the bu
tton style (modern or classic).
   virtual void         SetStyle(const char *style);/// Set the
button style (modern or classic).

   virtual void         SavePrimitive(std::ostream &out, Option_
t *option = "");
/// Save a button widget as a C++ statement(s) on output stream
out.

   GContext_t GetNormGC() const { return fNormGC; }

   virtual void Pressed()  { Emit("Pressed()"); }   // *SIGNAL*
   virtual void Released() { Emit("Released()"); }  // *SIGNAL*
```

```
   virtual void Clicked()  { Emit("Clicked()"); }   // *SIGNAL*
   virtual void Toggled(Bool_t on) { Emit("Toggled(Bool_t)", on)
; }  // *SIGNAL*
```

TGTextButton

```
   static FontStruct_t GetDefaultFontStruct();/// Return default
 font structure.

   TGTextButton(const TGWindow *p, TGHotString *s, Int_t id = -1
,
                GContext_t norm = GetDefaultGC()(),
                FontStruct_t font = GetDefaultFontStruct(),
                UInt_t option = kRaisedFrame | kDoubleBorder);
/// Create a text button widget. The hotstring will be adopted a
nd deleted
/// by the text button.

   TGTextButton(const TGWindow *p = 0, const char *s = 0, Int_t
id = -1,
                GContext_t norm = GetDefaultGC()(),
                FontStruct_t font = GetDefaultFontStruct(),
                UInt_t option = kRaisedFrame | kDoubleBorder);
/// Create a text button widget.

   TGTextButton(const TGWindow *p, const char *s, const char *cm
d,
                Int_t id = -1, GContext_t norm = GetDefaultGC()(
),
                FontStruct_t font = GetDefaultFontStruct(),
                UInt_t option = kRaisedFrame | kDoubleBorder);
/// Create a text button widget and set cmd string at same time.

   virtual ~TGTextButton();/// Delete a text button widget.

   virtual TGDimension GetDefaultSize() const;/// returns defaul
t size

   virtual Bool_t     HandleKey(Event_t *event);
```

```
/// Handle key event. This function will be called when the hotk
ey is hit.

   const TGHotString *GetText() const { return fLabel; }
   virtual const char *GetTitle() const { return fLabel->Data();
 }
   TString            GetString() const { return TString(fLabel-
>GetString()); }
   virtual void       SetTextJustify(Int_t tmode);
/// Set text justification. Mode is an OR of the bits:
/// kTextTop, kTextBottom, kTextLeft, kTextRight, kTextCenterX a
nd
/// kTextCenterY.

   Int_t GetTextJustify() const { return fTMode; }
   virtual void       SetText(TGHotString *new_label);/// Set ne
w button text.
   virtual void       SetText(const TString &new_label);/// Set
new button text.
   virtual void       SetTitle(const char *label) { SetText(labe
l); }
   virtual void       SetFont(FontStruct_t font, Bool_t global =
 kFALSE);
/// Changes text font.
/// If global is kTRUE font is changed globally, otherwise - loc
ally.

   virtual void       SetFont(const char *fontName, Bool_t globa
l = kFALSE);
/// Changes text font specified by name.
/// If global is true color is changed globally, otherwise - loc
ally.

   virtual void       SetTextColor(Pixel_t color, Bool_t global
= kFALSE);
/// Changes text color.
/// If global is true color is changed globally, otherwise - loc
ally.

   virtual void       SetForegroundColor(Pixel_t fore) { SetText
```

```
Color(fore); }
   Bool_t              HasOwnFont() const;
/// Returns kTRUE if text attributes are unique,
/// returns kFALSE if text attributes are shared (global).

   void                SetWrapLength(Int_t wl) { fWrapLength = wl
; Layout(); }
   Int_t               GetWrapLength() const { return fWrapLength
; }
   void                SetMargins(Int_t left=0, Int_t right=0, In
t_t top=0, Int_t bottom=0)
                         { fMLeft = left; fMRight = right; fMTop
= top; fMBottom = bottom; }

   virtual void        SetLeftMargin(Int_t val)   { fMLeft = val;
 }
   virtual void        SetRightMargin(Int_t val)  { fMRight = val
; }
   virtual void        SetTopMargin(Int_t val)    { fMTop = val;
}
   virtual void        SetBottomMargin(Int_t val) { fMBottom = va
l; }

   Int_t               GetLeftMargin() const { return fMLeft; }
   Int_t               GetRightMargin() const { return fMRight; }
   Int_t               GetTopMargin() const { return fMTop; }
   Int_t               GetBottomMargin() const { return fMBottom;
 }

   void                ChangeText(const char *title)  { SetTitle(
title); } //*MENU*icon=bld_rename.png*

   FontStruct_t GetFontStruct() const { return fFontStruct; }

   virtual void        Layout();/// layout text button
   virtual void        SavePrimitive(std::ostream &out, Option_t
*option = "");
/// Save a text button widget as a C++ statement(s) on output st
ream out.
```

TGPictureButton

```
   TGPictureButton(const TGWindow *p, const TGPicture *pic, Int_
t id = -1,
                   GContext_t norm = GetDefaultGC()(),
                   UInt_t option = kRaisedFrame | kDoubleBorder)
;
/// Create a picture button widget. The picture is not adopted a
nd must
/// later be freed by the user once the picture button is delete
d (a single
/// picture reference might be used by other buttons).

   TGPictureButton(const TGWindow *p, const TGPicture *pic, const
 char *cmd,
                   Int_t id = -1, GContext_t norm = GetDefaultGC
()(),
                   UInt_t option = kRaisedFrame | kDoubleBorder)
;
/// Create a picture button widget and set action command. The p
icture is
/// not adopted and must later be freed by the user once the pic
ture button
/// is deleted (a single picture reference might be used by other

/// buttons).

   TGPictureButton(const TGWindow *p = 0, const char* pic = 0, I
nt_t id = -1,
                   GContext_t norm = GetDefaultGC()(),
                   UInt_t option = kRaisedFrame | kDoubleBorder)
;
/// Create a picture button. Where pic is the file name of the p
icture.

   virtual ~TGPictureButton();/// Destructor.

   virtual void     SetPicture(const TGPicture *new_pic);
/// Change a picture in a picture button. The picture is not ado
pted and
```

```
/// must later be freed by the user once the picture button is d
eleted
/// (a single picture reference might be used by other buttons).

   virtual void     SetDisabledPicture(const TGPicture *pic);///
 Changes disabled picture.
   const TGPicture *GetPicture() const { return fPic; };
   const TGPicture *GetDisabledPicture() const { return fPicD; }
;
   virtual void     SavePrimitive(std::ostream &out, Option_t *o
ption = "");
/// Save a picture button widget as a C++ statement(s) on output
 stream out.
```

TGCheckButton

```
   static FontStruct_t  GetDefaultFontStruct();/// Return defaul
t font structure.
   static const TGGC   &GetDefaultGC();/// Return default graphi
cs context.

   TGCheckButton(const TGWindow *p, TGHotString *s, Int_t id = -1
,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t option = 0);
/// Create a check button widget. The hotstring will be adopted
and deleted
/// by the check button.

   TGCheckButton(const TGWindow *p = 0, const char *s = 0, Int_t
 id = -1,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t option = 0);/// Create a check button wi
dget.

   TGCheckButton(const TGWindow *p, const char *s, const char *c
md, Int_t id = -1,
```

```
                    GContext_t norm = GetDefaultGC()(),
                    FontStruct_t font = GetDefaultFontStruct(),
                    UInt_t option = 0);/// Create a check button wi
dget.

   virtual ~TGCheckButton();/// Delete a check button.

   virtual TGDimension GetDefaultSize() const;/// default size

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event.
   virtual Bool_t HandleKey(Event_t *event);/// Handle key event
. This function will be called when the hotkey is hit.
   virtual Bool_t HandleCrossing(Event_t *event);/// Handle mous
e crossing event.
   virtual Bool_t IsToggleButton() const { return kTRUE; }
   virtual Bool_t IsOn() const { return fState == kButtonDown; }
   virtual Bool_t IsDown() const { return fState == kButtonDown;
 }
   virtual Bool_t IsDisabledAndSelected() const { return ((fStat
e == kButtonDisabled) && fStateOn); }
   virtual void   SetDisabledAndSelected(Bool_t);
/// Set the state of a check button to disabled and either on or
 off.

   virtual void   SetState(EButtonState state, Bool_t emit = kFA
LSE);/// Set check button state.
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a check button widget as a C++ statement(s) on output s
tream out.
```

TGRadioButton

```
   static FontStruct_t  GetDefaultFontStruct();/// Return defaul
t font structure.
   static const TGGC   &GetDefaultGC();/// Return default graphi
cs context.
```

```
   TGRadioButton(const TGWindow *p, TGHotString *s, Int_t id = -1
,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t option = 0);
/// Create a radio button widget. The hotstring will be adopted
and deleted
/// by the radio button.

   TGRadioButton(const TGWindow *p = 0, const char *s = 0, Int_t
 id = -1,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t option = 0);/// Create a radio button wi
dget.
   TGRadioButton(const TGWindow *p, const char *s, const char *c
md, Int_t id = -1,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t option = 0);/// Create a radio button wi
dget.
   virtual ~TGRadioButton();/// Delete a radio button.

   virtual TGDimension GetDefaultSize() const;/// default size

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event.
   virtual Bool_t HandleKey(Event_t *event);
   /// Handle key event. This function will be called when the h
otkey is hit.

   virtual Bool_t HandleCrossing(Event_t *event);/// Handle mous
e crossing event.
   virtual void   SetState(EButtonState state, Bool_t emit = kFA
LSE);/// Set radio button state.
   virtual void   SetDisabledAndSelected(Bool_t);
/// Set the state of a radio button to disabled and either on or
 off.

   virtual Bool_t IsToggleButton() const { return kTRUE; }
```

```
   virtual Bool_t IsExclusiveToggle() const { return kTRUE; }
   virtual Bool_t IsOn() const { return fStateOn; }
   virtual Bool_t IsDown() const { return fStateOn; }
   virtual Bool_t IsDisabledAndSelected() const { return ((fStat
e == kButtonDisabled) && fStateOn); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a radio button widget as a C++ statement(s) on output s
tream out.
```

TGSplitButton

```
   TGSplitButton(const TGWindow *p, TGHotString *menulabel,
                 TGPopupMenu *popmenu, Bool_t split = kTRUE,
                 Int_t id = -1, GContext_t norm = GetDefaultGC()(
),
                 FontStruct_t fontstruct = GetDefaultFontStruct()
,
                 UInt_t option = kRaisedFrame | kDoubleBorder);
/// Create a menu button widget. The hotstring will be adopted a
nd
/// deleted by the menu button. This constructior creates a
/// menubutton with a popup menu attached that appears when the
/// button for it is clicked. The popup menu is adopted.

   virtual ~TGSplitButton();/// Delete a split button widget.

   virtual TGDimension GetDefaultSize() const ;/// returns defau
lt size

   virtual void   SetText(TGHotString *new_label);/// Set new bu
tton text.
   virtual void   SetText(const TString &new_label);/// Set new
button text.
   virtual void   SetFont(FontStruct_t font, Bool_t global = kFA
LSE);
/// Changes text font.
/// If global is kTRUE font is changed globally, otherwise - loc
ally.
```

```
   virtual void   SetFont(const char *fontName, Bool_t global =
kFALSE);
/// Changes text font specified by name.
/// If global is true color is changed globally, otherwise - loc
ally.

   virtual void   SetMBState(EButtonState state);/// Set the sta
te of the Menu Button part
   virtual void   SetSplit(Bool_t split);/// Set the split statu
s of a button.
   Bool_t         IsSplit() { return fSplit; }
   virtual Bool_t HandleButton(Event_t *event);/// Handle button
 events.
   virtual Bool_t HandleCrossing(Event_t *event);/// Handle mous
e crossing event.
   virtual Bool_t HandleKey(Event_t *event);
/// Handle key event. This function will be called when the hotk
ey is hit.

   virtual Bool_t HandleMotion(Event_t *event);/// Handle a moti
on event in a TGSplitButton.
   virtual void   Layout();/// layout text button

   virtual void MBPressed()  { Emit("MBPressed()"); }   // *SIGN
AL*
   virtual void MBReleased() { Emit("MBReleased()"); }  // *SIGN
AL*
   virtual void MBClicked()  { Emit("MBClicked()"); }   // *SIGN
AL*
   virtual void ItemClicked(Int_t id) { Emit("ItemClicked(Int_t)"
, id); } // *SIGNAL*

   // Slots
   void HandleMenu(Int_t id) ;/// Handle a menu item activation.
```

code

```
// Button
#include "TGClient.h"
#include "TGButton.h"
```

```
// TGTextButton
// 按钮，按钮上有字，字可改变
TGTextButton     *fStart;
fStart = new TGTextButton(frame, "&Start");
// fStart = new TGTextButton(frame, "&Start the \n software""and
 go it");
// fStart->Resize(300, 200);//设置大小
// fStart->ChangeOptions(fStart->GetOptions() | kFixedSize);
ULong_t yellow;
gClient->GetColorByName("yellow", yellow);
fStart->ChangeBackground(yellow);//button background will be set
 to yellow
fStart->Connect("Clicked()", "MyMainFrame", this, "ChangeStartLa
bel()");
frame->AddFrame(fStart, new TGLayoutHints(kLHintsTop | kLHintsEx
pandX,3, 2, 2, 2));
fStart->SetToolTipText("Click to toggle the button label (Start/
Stop)");//鼠标放上去显示的信息
void ChangeStartLabel()
{
  fStart->SetState(kButtonDown);
  if (!start) {
     fStart->SetText("&Stop");
     start = kTRUE;
  } else {
     fStart->SetText("&Start");
     start = kFALSE;
  }
  fStart->SetState(kButtonUp);
}
```

```
// TGCheckButton
// 多选按钮
TGVButtonGroup *fButtonGroup;  // Button group
fButtonGroup = new TGVButtonGroup(frame, "My Button Group");
fButtonGroup->SetTitlePos(TGGroupFrame::kCenter);
frame->AddFrame(fButtonGroup, new TGLayoutHints(kLHintsCenterX|k
LHintsCenterY,1, 1, 1, 1));
// TGGroupFrame *fButtonGroup = new TGGroupFrame(frame, "Enable/
Disable");
// fButtonGroup->SetTitlePos(TGGroupFrame::kCenter);
// frame->AddFrame(fButtonGroup, new TGLayoutHints(kLHintsExpand
X ));
TGCheckButton *fCheckb[4];
fCheckb[0] = new TGCheckButton(fButtonGroup, new TGHotString("CB
 1"),IDs);//Widget IDs ，每个的唯一编号，0，1，2，3……
fCheckb[1] = new TGCheckButton(fButtonGroup, new TGHotString("CB
 2"),IDs);
fCheckb[2] = new TGCheckButton(fButtonGroup, new TGHotString("CB
 3"),IDs);
fCheckb[3] = new TGCheckButton(fButtonGroup, new TGHotString("CB
 4"),IDs);
fCheckb[0]->SetOn();//Default state
// fCheckb[0]->->SetState(kButtonDown);//设置状态为选上
fCheckb[0]->GetState();//获得当前状态，0、1
// fCheckb[0]->Connect("Toggled(Bool_t)", "MyButtonTest", this,
"SetGroupEnabled(Bool_t)");
fButtonGroup->Show();
fButtonGroup->SetState(kTRUE);//是否开启，开启才可以选，不开启是灰色的
```

```
// TGRadioButton
// 单选按钮
TGVButtonGroup      *fButtonGroup;  // Button group
TGRadioButton *fRadiob[2];
fButtonGroup = new TGVButtonGroup(frame, "My Button Group");
fButtonGroup->SetTitlePos(TGGroupFrame::kCenter);
fRadiob[0] = new TGRadioButton(fButtonGroup, new TGHotString("RB
 1"),IDs);
fRadiob[1] = new TGRadioButton(fButtonGroup, new TGHotString("RB
 2"),IDs);
fRadiob[1]->SetOn();//Default state
fButtonGroup->Show();
fButtonGroup->SetState(kTRUE);//是否开启，开启才可以选，不开启是灰色的
fButtonGroup->SetRadioButtonExclusive(kTRUE);//???
frame->AddFrame(fButtonGroup, new TGLayoutHints(kLHintsCenterX|k
LHintsCenterY,1, 1, 1, 1));
```

```
// TGPictureButton
// 按钮是图片
```

```
// TGSplitButton
//
```

example

TGButtonGroup

TGButtonGroup, TGVButtonGroup and TGHButtonGroup
This header defines button group frames.

The TGButtonGroup widget organizes TGButton widgets in a group.

A button group widget makes it easier to deal with groups of buttons. A button in a button group is associated with a unique identifier. The button group emits a Clicked() signal with this identifier when the button is clicked. Thus, a button group is an ideal solution when you have several similar buttons and want to connect all their Clicked() signals, for example, to one slot.

An exclusive button group switches off all toggle buttons except the one that was clicked. A button group is by default non-exclusive. All radio buttons that are inserted, will be mutually exclusive even if the button group is non-exclusive.

There are two ways of using a button group:

The button group is a parent widget of a number of buttons, i.e. the button group is the parent argument in the button constructor. The buttons are assigned identifiers 1, 2, 3 etc. in the order they are created or you can specify button id in the button constructor. A TGButtonGroup can display a frame and a title because it inherits from TGGroupFrame.

NOTE: there is no need to call AddFrame() since the buttons are automatically added with a default layout hint to their parent, i.e. the buttongroup. To override the default layout hints use the SetLayoutHints() method.

```
 ButtonGroup Signals:


   Pressed(Int_t id)  -->  is emitted when a button in the group
 is
                           pressed down. The id argument is the

                           button's identifier.

   Released(Int_t id) -->  is emitted when a button in the group
 is
                           released. The id argument is the butt
on's
                           identifier.

   Clicked(Int_t id)  -->  is emitted when a button in the group
 is
                           clicked. The id argument is the butto
n's
                           identifier.
```

The TGHButtonGroup widget organizes TGButton widgets in a group with one horizontal row. TGHButtonGroup is a convenience class that offers a thin layer on top of TGButtonGroup. It inherits from TGButtonGroup.

The TGVButtonGroup widget organizes TGButton widgets in a group with one vertical column. TGVButtonGroup is a convenience class that offers a thin layer on top of TGButtonGroup. It inherits from TGButtonGroup.

TGButtonGroup 继承 TGGroupFrame , friend TGButton

TGVButtonGroup 继承 TGButtonGroup

TGHButtonGroup 继承 TGButtonGroup

class

TGButtonGroup

```
   TGButtonGroup(const TGWindow *parent = 0,
                 const TString &title = "",
                 UInt_t options = kChildFrame | kVerticalFrame,
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 Pixel_t back = GetDefaultFrameBackground());
/// Constructor. Layout 1 row or 1 column.

   TGButtonGroup(const TGWindow *parent,
                 UInt_t r, UInt_t c, Int_t s = 0, Int_t h = 0 ,
                 const TString &title = "",
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 Pixel_t back = GetDefaultFrameBackground());
/// Constructor. Layout defined by TGMatrixLayout:
///    r = number of rows
///    c = number of columns
///    s = interval between frames
///    h = layout hints

   virtual ~TGButtonGroup();/// Destructor, we do not delete the
 buttons.

   virtual void Pressed(Int_t id)  { Emit("Pressed(Int_t)",id);
}   //*SIGNAL*
   virtual void Released(Int_t id) { Emit("Released(Int_t)",id);
}   //*SIGNAL*
   virtual void Clicked(Int_t id)  { Emit("Clicked(Int_t)",id);
}   //*SIGNAL*

   virtual void ButtonPressed();
/// This slot is activated when one of the buttons in the group
emits the
/// Pressed() signal.

   virtual void ButtonReleased();
/// This slot is activated when one of the buttons in the group
emits the
```

```
/// Released() signal.

   virtual void ButtonClicked();
/// This slot is activated when one of the buttons in the group
emits the
/// Clicked() signal.

   virtual void ReleaseButtons();
/// This slot is activated when one of the buttons in the
/// exclusive group emits the Pressed() signal.

   Bool_t IsEnabled() const { return fState; }
   Bool_t IsExclusive() const { return fExclGroup; }
   Bool_t IsRadioButtonExclusive() const  { return fRadioExcl; }
   Bool_t IsBorderDrawn() const { return fDrawBorder; }
   Int_t  GetCount() const { return fMapOfButtons->GetSize(); }
   Int_t  GetId(TGButton *button) const;
/// Finds and returns the id of the button.
/// Returns -1 if the button is not a member of this group.

   virtual void SetExclusive(Bool_t flag = kTRUE);
/// Sets the button group to be exclusive if enable is kTRUE,
/// or to be non-exclusive if enable is kFALSE.
/// An exclusive button group switches off all other toggle butt
ons when
/// one is switched on. This is ideal for groups of radio-buttons

/// A non-exclusive group allow many buttons to be switched on a
t the same
/// time. The default setting is kFALSE.

   virtual void SetRadioButtonExclusive(Bool_t flag = kTRUE);
/// If enable is kTRUE, this button group will treat radio butto
ns as
/// mutually exclusive, and other buttons according to IsExclusi
ve().
/// This function is called automatically whenever a TGRadioButt
on
/// is inserted, so you should normally never have to call it.
```

```
   virtual void SetState(Bool_t state = kTRUE);
/// Sets the state of all the buttons in the group to enable or
disable.

   virtual void SetBorderDrawn(Bool_t enable = kTRUE);/// Makes
border to be visible/invisible.
   virtual void SetButton(Int_t id, Bool_t down = kTRUE);
/// Sets the button with id to be on/down, and if this is an
/// exclusive group, all other button in the group to be off/up.

   virtual void SetTitle(TGString *title);/// Set or change titl
e.
   virtual void SetTitle(const char *title);/// Set or change ti
tle.

   virtual Int_t     Insert(TGButton *button, int id = -1);
/// Inserts a button with the identifier id into the button grou
p.
/// Returns the button identifier.
/// It is not necessary to manually insert buttons that have thi
s button
/// group as their parent widget. An exception is when you want
custom
/// identifiers instead of the default 1, 2, 3 etc.
/// The button is assigned the identifier id or an automatically
/// generated identifier.  It works as follows: If id > 0, this
/// identifier is assigned.  If id == -1 (default), the identifi
er is
/// equal to the number of buttons in the group+1.  If id is any
 other
/// negative integer, for instance -2, a unique identifier (nega
tive
/// integer <= -2) is generated.
/// Inserting several buttons with id = -1 assigns the identifie
rs 1,
/// 2, 3, etc.

   virtual void      Remove(TGButton *button);/// Removes a butt
on from the button group.
   virtual TGButton *Find(Int_t id) const;
```

```
/// Finds and returns a pointer to the button with the specified
/// identifier id. Returns null if the button was not found.

   virtual TGButton *GetButton(Int_t id) const { return Find(id)
; }
   virtual void      Show();/// Show group of buttons.
   virtual void      Hide();/// Hide group of buttons.
   virtual void      DrawBorder();
/// Draw border of around the group frame.
/// if frame is kRaisedFrame  - a frame border is of "wall style
",
/// otherwise of "groove style".

   virtual void      SetLayoutHints(TGLayoutHints *l, TGButton *
button = 0);
/// Set layout hints for the specified button or if button=0 for
 all
/// buttons.

   virtual void      SavePrimitive(std::ostream &out, Option_t *
option = "");
/// Save a button group widget as a C++ statement(s) on output s
tream out.
```

TGVButtonGroup

```
   TGVButtonGroup(const TGWindow *parent,
                  const TString &title = "",
                  GContext_t norm = GetDefaultGC()(),
                  FontStruct_t font = GetDefaultFontStruct(),
                  Pixel_t back = GetDefaultFrameBackground()) :
      TGButtonGroup(parent, title, kChildFrame | kVerticalFrame,
                    norm, font, back) { }

   virtual ~TGVButtonGroup() { }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a button group widget as a C++ statement(s) on output s
tream out.
```

TGHButtonGroup

```
   TGHButtonGroup(const TGWindow *parent,
                  const TString &title = "",
                  GContext_t norm = GetDefaultGC()(),
                  FontStruct_t font = GetDefaultFontStruct(),
                  Pixel_t back = GetDefaultFrameBackground()) :
      TGButtonGroup(parent, title, kChildFrame | kHorizontalFram
e,
                    norm, font, back) { }

   virtual ~TGHButtonGroup() { }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a button group widget as a C++ statement(s) on output s
tream out.
```

code

// Example: // // // // // vertical frame without border and title // // TGVButtonGroup *bg = new TGVButtonGroup(main_frame); // // // // // create text button with id=1 // // TGTextButton* button1 = new TGTextButton(bg,"some text"); // // // // // create

another text button with id=2 // // TGTextButton *button2 = new TGTextButton(bg,"another text"); // // // // // map all buttons // // bg->Show(); //

example

TGCanvas

A TGCanvas is a frame containing two scrollbars (horizontal and vertical) and a viewport. The viewport acts as the window through which we look at the contents of the container frame.

A TGContainer frame manages a content area. It can display and control a hierarchy of multi-column items, and provides the ability to add new items at any time. By default it doesn't map subwindows which are items of the container. In this case subwindow must provide DrawCopy method, see for example TGLVEntry class.
It is also possible to use option which allow to map subwindows.
This option has much slower drawing speed in case of more than 1000 items placed in container. To activate this option the fMapSubwindows data member must be set to kTRUE (for example TTVLVContainer class)

The TGContainer class can handle the keys:

- F7, Ctnrl-F - activate search dialog
- F3, Ctnrl-G - continue search
- End - go to the last item in container
- Home - go to the first item in container
- PageUp,PageDown,arrow keys - navigate inside container
- Return/Enter - equivalent to double click of the mouse button
- Contrl-A - select/activate all items.
- Space - invert selection.

TGContainer 继承 TGCompositeFrame , friend TGViewPort , TGCanvas , TGContainerKeyboardTimer , TGContainerScrollTimer , TGListView

TGViewPort 继承 TGCompositeFrame

TGCanvas 继承 TGFrame

class

TGContainer

```
   TGContainer(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1
,
               UInt_t options = kSunkenFrame,
               Pixel_t back = GetDefaultFrameBackground());
/// Create a canvas container. This is the (large) frame that co
ntains
/// all the list items. It will be shown through a TGViewPort (w
hich is
/// created by the TGCanvas).

   TGContainer(TGCanvas *p,UInt_t options = kSunkenFrame,
               Pixel_t back = GetDefaultFrameBackground());
/// Create a canvas container. This is the (large) frame that co
ntains
/// all the list items. It will be shown through a TGViewPort (w
hich is
/// created by the TGCanvas).

   virtual ~TGContainer();/// Delete canvas container.

   virtual void DrawRegion(Int_t x, Int_t y, UInt_t w, UInt_t h)
;
/// Draw a region of container in viewport.
/// x, y, w, h are position and dimension of area to be
/// redrawn in viewport coordinates.

   virtual void ClearViewPort();/// Clear view port and redraw f
ull content
   virtual void Associate(const TGWindow *w) { fMsgWindow = w; }
   virtual void AdjustPosition();/// Move content to position of
 highlighted/activated frame.
   virtual void SetPagePosition(const TGPosition& pos);/// Set p
age position.
   virtual void SetPagePosition(Int_t x, Int_t y);/// Set page p
osition.
   virtual void SetPageDimension(const TGDimension& dim);/// Set
 page dimension.
   virtual void SetPageDimension(UInt_t w, UInt_t h);/// Set pag
e dimension.
```

```
   virtual void RemoveAll();/// Remove all items from the contai
ner.
   virtual void RemoveItem(TGFrame *item);/// Remove item from c
ontainer.
   virtual void Layout();/// Layout container entries.

   TGCanvas         *GetCanvas() const { return fCanvas; }
   const TGWindow   *GetMessageWindow() const { return fMsgWindo
w; }
   virtual TGPosition   GetPagePosition() const;/// Returns page
 position.
   virtual TGDimension  GetPageDimension() const;/// Returns pag
e dimension.

   virtual Int_t  NumSelected() const { return fSelected; }
   virtual Int_t  NumItems() const { return fTotal; }
   virtual TGFrameElement *FindFrame(Int_t x,Int_t y,Bool_t excl
ude=kTRUE);
/// Find frame located int container at position x,y.

   virtual TGFrame        *FindFrameByName(const char *name);///
 Find frame by name.
   virtual TGHScrollBar *GetHScrollbar() const;/// returns point
er to hor. scroll bar
   virtual TGVScrollBar *GetVScrollbar() const;/// returns point
er to vert. scroll bar
   virtual void SetHsbPosition(Int_t newPos);/// set new hor. po
sition
   virtual void SetVsbPosition(Int_t newPos);/// Set position of
 vertical scrollbar.
   virtual void LineUp(Bool_t select = kFALSE);/// Make current
position first line in window by scrolling up.
   virtual void LineDown(Bool_t select = kFALSE);/// Move one li
ne down.
   virtual void LineLeft(Bool_t select = kFALSE);/// Move curren
t position one column left.
   virtual void LineRight(Bool_t select = kFALSE);/// Move curre
nt position one column right.
   virtual void PageUp(Bool_t select = kFALSE);/// Move  positio
n one page up.
```

```
   virtual void PageDown(Bool_t select = kFALSE);/// Move positi
on one page down.
   virtual void Home(Bool_t select = kFALSE);/// Move to upper-l
eft corner of container.
   virtual void End(Bool_t select = kFALSE);/// Move to the bott
om-right corner of container.
   virtual void Search(Bool_t close = kTRUE);/// Invokes search
dialog. Looks for item with the entered name.
   virtual void *FindItem(const TString& name,
                          Bool_t direction = kTRUE,
                          Bool_t caseSensitive = kTRUE,
                          Bool_t subString = kFALSE);

   virtual const TGFrame *GetNextSelected(void **current);
/// Return the next selected item. If the "current" pointer is 0
, the first
/// selected item will be returned.

   virtual TGFrame *GetLastActive() const { return fLastActiveEl
 ? fLastActiveEl->fFrame : 0; }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a canvas container as a C++ statement(s) on output stre
am out.

   virtual Bool_t HandleDNDFinished() { fBdown = kFALSE; return
kTRUE; }
   virtual Bool_t HandleExpose(Event_t *event);/// Handle expose
 events. Do not use double buffer.
   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in container.
   virtual Bool_t HandleDoubleClick(Event_t *event);/// Handle d
ouble click mouse event.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion events.
   virtual Bool_t HandleKey(Event_t *event);/// The key press ev
ent handler converts a key press to some line editor action.

   const TGPicture *GetObjPicture(TGFrame *f);
/// Retrieve icons associated with class "name". Association is
```

```
made
/// via the user's ~/.root.mimes file or via $ROOTSYS/etc/root.m
imes.

   virtual void SetDragPixmap(const TGPicture *pic);/// Set drag
 window pixmaps and hotpoint.

   virtual void SelectAll();                    //*SIGNAL*
/// Select all items in the container.
/// SelectAll() signal emitted.

   virtual void UnSelectAll();                  //*SIGNAL*  ///
Unselect all items in the container.
   virtual void InvertSelection();              //*SIGNAL*
/// Invert the selection, all selected items become unselected a
nd vice versa.

   virtual void ReturnPressed(TGFrame*);        //*SIGNAL*
/// Signal emitted when Return/Enter key pressed.
/// It's equivalent to "double click" of mouse button.

   virtual void SpacePressed(TGFrame*);         //*SIGNAL*
/// Signal emitted when space key pressed.
/// Pressing space key inverts selection.

   virtual void KeyPressed(TGFrame*, UInt_t keysym, UInt_t mask)
; //*SIGNAL*
/// Signal emitted when keyboard key pressed
/// frame - activated frame
/// keysym - defined in "KeySymbols.h"
/// mask - modifier key mask, defined in "GuiTypes.h"
/// const Mask_t kKeyShiftMask   = BIT(0);
/// const Mask_t kKeyLockMask    = BIT(1);
/// const Mask_t kKeyControlMask = BIT(2);
/// const Mask_t kKeyMod1Mask    = BIT(3);   // typically the Al
t key
/// const Mask_t kButton1Mask    = BIT(8);
/// const Mask_t kButton2Mask    = BIT(9);
/// const Mask_t kButton3Mask    = BIT(10);
/// const Mask_t kButton4Mask    = BIT(11);
```

```
/// const Mask_t kButton5Mask    = BIT(12);
/// const Mask_t kAnyModifier    = BIT(15);

   virtual void OnMouseOver(TGFrame*);          //*SIGNAL*  ///
Signal emitted when pointer is over entry.
   virtual void CurrentChanged(Int_t x,Int_t y);//*SIGNAL*  ///
Emit signal when current position changed.
   virtual void CurrentChanged(TGFrame* f);     //*SIGNAL*  ///
Emit signal when current selected frame changed.
   virtual void Clicked(TGFrame *f, Int_t btn); //*SIGNAL*  ///
Emit Clicked() signal.
   virtual void DoubleClicked(TGFrame *f, Int_t btn);  //*SIGNAL
*  /// Emit DoubleClicked() signal.
   virtual void DoubleClicked(TGFrame *f, Int_t btn, Int_t x, In
t_t y); //*SIGNAL*  /// Emit DoubleClicked() signal.
   virtual void Clicked(TGFrame *f, Int_t btn, Int_t x, Int_t y)
;       //*SIGNAL*  /// Emit Clicked() signal.
```

TGViewPort

```
   TGViewPort(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
              UInt_t options = kChildFrame,
              Pixel_t back = GetDefaultFrameBackground());/// Cr
eate a viewport object.

   TGFrame *GetContainer() const { return fContainer; }
   void SetContainer(TGFrame *f);
/// Add container frame to the viewport. We must make sure that
the added
/// container is at least a TGCompositeFrame (TGCanvas::AddFrame
 depends
/// on it).

   virtual void DrawBorder() { };
   virtual void Layout() { }
   virtual TGDimension GetDefaultSize() const { return TGDimensi
on(fWidth, fHeight); }

   virtual void SetHPos(Int_t xpos);/// Moves content of contain
er frame in horizontal direction.
   virtual void SetVPos(Int_t ypos);/// Moves content of contain
er frame in vertical direction.
   void SetPos(Int_t xpos, Int_t ypos);/// Goto new position.

   Int_t GetHPos() const { return fX0; }
   Int_t GetVPos() const { return fY0; }
   virtual Bool_t HandleConfigureNotify(Event_t *event);/// Hand
le resize events.
```

TGCanvas

```
   enum { kCanvasNoScroll         = 0,
          kCanvasScrollHorizontal = BIT(0),
          kCanvasScrollVertical   = BIT(1),
          kCanvasScrollBoth       = (kCanvasScrollHorizontal | k
CanvasScrollVertical)
   };

   TGCanvas(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
```

```
            UInt_t options = kSunkenFrame | kDoubleBorder,
            Pixel_t back = GetDefaultFrameBackground());/// Crea
te a canvas object.

   virtual ~TGCanvas();/// Delete canvas.

   TGFrame      *GetContainer() const { return fVport->GetContai
ner(); }
   TGViewPort   *GetViewPort() const { return fVport; }
   TGHScrollBar *GetHScrollbar() const { return fHScrollbar; }
   TGVScrollBar *GetVScrollbar() const { return fVScrollbar; }

   virtual void  AddFrame(TGFrame *f, TGLayoutHints *l = 0);
/// Adding a frame to a canvas is actually adding the frame to t
he
/// viewport container. The viewport container must be at least a

/// TGCompositeFrame for this method to succeed.

   virtual void  SetContainer(TGFrame *f) { fVport->SetContainer
(f); }
   virtual void  MapSubwindows();
   virtual void  DrawBorder();/// Draw canvas border.
   virtual void  Layout();
/// Create layout for canvas. Depending on the size of the conta
iner
/// we need to add the scrollbars.

   virtual void  ClearViewPort();/// Clear view port and redraw
content.
   virtual Int_t GetHsbPosition() const;/// Set position of hori
zontal scrollbar.
   virtual Int_t GetVsbPosition() const;/// Get position of vert
ical scrollbar.
   virtual void  SetHsbPosition(Int_t newPos);/// Get position o
f horizontal scrollbar.
   virtual void  SetVsbPosition(Int_t newPos);/// Set position o
f vertical scrollbar.
   void          SetScrolling(Int_t scrolling);
/// Set scrolling policy. Use values defined by the enum: kCanva
```

```
sNoScroll,
/// kCanvasScrollHorizontal, kCanvasScrollVertical, kCanvasScrol
lBoth.

   Int_t         GetScrolling() const { return fScrolling; }

   virtual TGDimension GetDefaultSize() const { return TGDimensi
on(fWidth, fHeight); }
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);/// Handle message generated by the canvas scrollbars.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a canvas widget as a C++ statement(s) on output stream
out.
```

code

example

TGClient

TGClient 继承 TObject, TQObject , friend TGCocoa

Window client. In client server windowing systems, like X11 this class is used to make the initial connection to the window server.

class

```
   TGClient(const char *dpyName = 0);
/// Create a connection with the display sever on host dpyName a
nd setup
/// the complete GUI system, i.e., graphics contexts, fonts, etc
. for all
/// widgets.

   virtual ~TGClient();/// Closing down client: cleanup and clos
e X connection.

   const TGWindow *GetRoot() const;
/// Returns current root (i.e. base) window. By changing the root

/// window one can change the window hierarchy, e.g. a top level
/// frame (TGMainFrame) can be embedded in another window.

   const TGWindow *GetDefaultRoot() const;
/// Returns the root (i.e. desktop) window. Should only be used
as parent
/// for frames that will never be embedded, like popups, message
 boxes,
/// etc. (like TGToolTips, TGMessageBox, etc.).

   void            SetRoot(TGWindow *root = 0);
/// Sets the current root (i.e. base) window. By changing the ro
ot
/// window one can change the window hierarchy, e.g. a top level
/// frame (TGMainFrame) can be embedded in another window.
```

```
   TGWindow       *GetWindowById(Window_t sw) const;
/// Find a TGWindow via its handle. If window is not found retur
n 0.

   TGWindow       *GetWindowByName(const char *name) const;
/// Find a TGWindow via its name (unique name used in TGWindow::
SavePrimitive).
/// If window is not found return 0.

   UInt_t       GetDisplayWidth() const;/// Get display width.
   UInt_t       GetDisplayHeight() const;/// Get display height.

   Bool_t       IsEditable() const { return fRoot != fDefaultRoo
t; }
   Bool_t       IsEditDisabled() const;/// Returns kTRUE if edit
/guibuilding is forbidden.
   void         SetEditDisabled(Bool_t on = kTRUE);/// If on is
kTRUE editting/guibuilding is forbidden.

   FontStruct_t GetFontByName(const char *name, Bool_t fixedDefa
ult = kTRUE) const;
/// Get a font by name. If font is not found, fixed font is retu
rned,
/// if fixed font also does not exist return 0 and print error.
/// The loaded font needs to be freed using TVirtualX::DeleteFon
t().
/// If fixedDefault is false the "fixed" font will not be substi
tuted
/// as fallback when the asked for font does not exist.

   Bool_t       GetColorByName(const char *name, Pixel_t &pixel)
const;
/// Get a color by name. If color is found return kTRUE and pixe
l is
/// set to the color's pixel value, kFALSE otherwise.

   Pixel_t      GetHilite(Pixel_t base_color) const;/// Return p
ixel value of hilite color based on base_color.
   Pixel_t      GetShadow(Pixel_t base_color) const;
```

```
/// Return pixel value of shadow color based on base_color.
/// Shadow is 60% of base_color intensity.

   void         FreeColor(Pixel_t color) const;/// Free color.
   void         ForceRedraw() { fForceRedraw = kTRUE; }
   void         NeedRedraw(TGWindow *w, Bool_t force = kFALSE);/
// Set redraw flags.
   void         CancelRedraw(TGWindow *w);
   void         RegisterWindow(TGWindow *w);/// Add a TGWindow t
o the clients list of windows.
   void         UnregisterWindow(TGWindow *w);/// Remove a TGWin
dow from the list of windows.
   void         RegisterPopup(TGWindow *w);
/// Add a popup menu to the list of popups. This list is used to
 pass
/// events to popup menus that are popped up over a transient wi
ndow which
/// is waited for (see WaitFor()).

   void         UnregisterPopup(TGWindow *w);/// Remove a popup
menu from the list of popups.
   void         AddUnknownWindowHandler(TGUnknownWindowHandler *
h);
/// Add handler for unknown (i.e. unregistered) windows.

   void         RemoveUnknownWindowHandler(TGUnknownWindowHandle
r *h);
/// Remove handler for unknown (i.e. unregistered) windows.

   void         AddIdleHandler(TGIdleHandler *h);/// Add handler
 for idle events.
   void         RemoveIdleHandler(TGIdleHandler *h);/// Remove h
andler for idle events.
   Bool_t       HandleInput();
/// Handles input from the display server. Returns kTRUE if one
or more
/// events have been processed, kFALSE otherwise.

   void         ProcessLine(TString cmd, Long_t msg, Long_t parm
1, Long_t parm2);
```

```
/// Execute string "cmd" via the interpreter. Before executing r
eplace
/// in the command string the token $MSG, $PARM1 and $PARM2 by m
sg,
/// parm1 and parm2, respectively. The function in cmd string mu
st accept
/// these as longs.

   void         WaitFor(TGWindow *w);/// Wait for window to be d
estroyed.
   void         WaitForUnmap(TGWindow *w);/// Wait for window to
 be unmapped.
   void         ResetWaitFor(TGWindow *w);/// reset waiting
   EGEventType  GetWaitForEvent() const  { return fWaitForEvent;
}
   Window_t     GetWaitForWindow() const { return fWaitForWindow
; }
   void         SetWaitForWindow(Window_t wid) {fWaitForWindow =
 wid;}
   Bool_t       ProcessEventsFor(TGWindow *w);
/// Like gSystem->ProcessEvents() but then only allow events for
 w to
/// be processed. For example to interrupt the processing and de
stroy
/// the window, call gROOT->SetInterrupt() before destroying the
 window.

   Bool_t       HandleEvent(Event_t *event);/// Handle a GUI eve
nt.
   Bool_t       HandleMaskEvent(Event_t *event, Window_t wid);
/// Handle masked events only if window wid is the window for wh
ich the
/// event was reported or if wid is a parent of the event window
. The not
/// masked event are handled directly. The masked events are:
/// kButtonPress, kButtonRelease, kKeyPress, kKeyRelease, kEnter
Notify,
/// kLeaveNotify, kMotionNotify.

   void         RegisteredWindow(Window_t w);      //*SIGNAL*
```

```
/// Emits a signal when a Window has been registered in TGClient.

/// Used in TRecorder.

   void          ProcessedEvent(Event_t *event, Window_t wid);
//*SIGNAL*
/// Emits a signal when an event has been processed.
/// Used in TRecorder.

   const TGResourcePool *GetResourcePool() const { return fResou
rcePool; }

   TGPicturePool   *GetPicturePool() const { return fPicturePool
; }
   const TGPicture *GetPicture(const char *name);
/// Get picture from the picture pool. Picture must be freed usi
ng
/// TGClient::FreePicture(). If picture is not found 0 is return
ed.

   const TGPicture *GetPicture(const char *name, UInt_t new_widt
h, UInt_t new_height);
/// Get picture with specified size from pool (picture will be s
caled if
/// necessary). Picture must be freed using TGClient::FreePictur
e(). If
/// picture is not found 0 is returned.

   void             FreePicture(const TGPicture *pic);/// Free p
icture resource.

   TGGCPool        *GetGCPool() const { return fGCPool; }
   TGGC            *GetGC(GCValues_t *values, Bool_t rw = kFALSE)
;
/// Get graphics context from the gc pool. Context must be freed
 via
/// TGClient::FreeGC(). If rw is true a new read/write-able GC
/// is returned, otherwise a shared read-only context is returne
d.
/// For historical reasons it is also possible to create directl
```

```
y a
/// TGGC object, but it is advised to use this new interface onl
y.

   void              FreeGC(const TGGC *gc);/// Free a graphics c
ontext.
   void              FreeGC(GContext_t gc);/// Free a graphics co
ntext.

   TGFontPool       *GetFontPool() const { return fFontPool; }
   TGFont           *GetFont(const char *font, Bool_t fixedDefaul
t = kTRUE);
/// Get a font from the font pool. Fonts must be freed via
/// TGClient::FreeFont(). Returns 0 in case of error or if font
/// does not exist. If fixedDefault is false the "fixed" font
/// will not be substituted as fallback when the asked for font
/// does not exist.

   TGFont           *GetFont(const TGFont *font);/// Get again sp
ecified font. Will increase its usage count.
   void              FreeFont(const TGFont *font);/// Free a font.


   UInt_t            GetStyle() const { return fStyle; }
   void              SetStyle(UInt_t newstyle) { fStyle = newstyl
e; }
   void              SetStyle(const char *style);/// Set the butt
on style (modern or classic).

   Colormap_t        GetDefaultColormap() const { return fDefault
Colormap; }
   TGMimeTypes      *GetMimeTypeList() const { return fMimeTypeLi
st; }

   THashList        *GetListOfWindows() const { return fWlist; }
   TList            *GetListOfPopups() const { return fPlist; }

   static TGClient *Instance();/// Returns global gClient (initi
alize graphics first, if not already done)
```

```
#ifndef __CINT__
#define gClient (TGClient::Instance())
#endif
```

code

example

TGColorDialog

The TGColorPalette is a widget showing an matrix of color cells. The colors can be set and selected.

The TGColorPick is a widget which allows a color to be picked from HLS space. It consists of two elements: a color map window from where the user can select the hue and saturation level of a color, and a slider to select color's lightness.

```
Selecting a color in these two widgets will generate the event:

kC_COLORSEL, kCOL_CLICK, widget id, 0.

and the signal:

ColorSelected(Pixel_t color)
```

The TGColorDialog presents a full featured color selection dialog.
It uses 2 TGColorPalette's and the TGColorPick widgets.

TGColorPalette 继承 TGFrame, TGWidget
Color palette widget

TGColorPick 继承 TGFrame, TGWidget
Color picker widget

TGColorDialog 继承 TGTransientFrame
Color selection dialog

class

TGColorPalette

```
   TGColorPalette(const TGWindow *p = 0, Int_t cols = 8, Int_t r
ows = 8, Int_t id = -1);
```

```
/// TGColorPalette widget: this is just a grid of color cells of
 the
/// specified size. Colors can be selected by clicking on them o
r by
/// using the arrow keys.

   virtual ~TGColorPalette();/// Destructor.

   virtual Bool_t HandleButton(Event_t *event);/// Handle button
 events in color palette
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion events in color palette.
   virtual Bool_t HandleKey(Event_t *event);/// Handle keyboard
events in color palette.

   virtual TGDimension GetDefaultSize() const
            { return TGDimension((fCw + 5) * fCols, (fCh + 5) *
fRows); }

   void    SetColors(Pixel_t colors[]);/// Set color entries in
color samples.
   void    SetColor(Int_t ix, Pixel_t color);/// Set color at in
dex ix of color entries.
   void    SetCurrentCellColor(Pixel_t color);/// Set current ce
ll color.

   void    SetCellSize(Int_t w = 20, Int_t h = 17);/// Set color
 cell size.

   Pixel_t GetColorByIndex(Int_t ix) const { return fPixels[ix];
 }
   Pixel_t GetCurrentColor() const;/// Return currently selected
 color value.

   virtual void ColorSelected(Pixel_t col = 0)
            { Emit("ColorSelected(Pixel_t)", col ? col : GetCurr
entColor()); }  //*SIGNAL*
`
```

TGColorPick

```
   TGColorPick(const TGWindow *p = 0, Int_t w = 1, Int_t h = 1,
Int_t id = -1);
/// TGColorPick constructor.
/// TGColorPick is a widget which allows a color to be picked fr
om HLS space.
/// It consists of two elements: a color map window from where t
he user can
/// select the hue and saturation level of a color, and a slider
 to select
/// color's lightness.

   virtual ~TGColorPick();/// TGColorPick destructor.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button events in color pick widget.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion events in color pick widget.

   void     SetColor(Pixel_t color);/// Position the slider curs
or on right color position.
   Pixel_t  GetCurrentColor() const { return fCurrentColor; }

   virtual  void ColorSelected(Pixel_t col = 0)
            { Emit("ColorSelected(Pixel_t)", col ? col : GetCurr
entColor()); }  //*SIGNAL*
`
```

TGColorDialog

```
   TGColorDialog(const TGWindow *p = 0, const TGWindow *m = 0, I
nt_t *retc = 0,
                 Pixel_t *color = 0, Bool_t wait = kTRUE);
/// Color selection dialog constructor.
/// The TGColorDialog presents a full featured color selection d
ialog.
/// It uses 2 TGColorPalette's and the TGColorPick widgets.

   virtual ~TGColorDialog();/// TGColorDialog destructor.

   TGColorPalette *GetPalette() const { return fPalette; }
   TGColorPalette *GetCustomPalette() const { return fCpalette;
}

   virtual void ColorSelected(Pixel_t); //*SIGNAL*  /// Emit sig
nal about selected color.
   virtual void AlphaColorSelected(ULong_t); //*SIGNAL*  /// Emi
t signal about selected alpha and color.
           void DoPreview();/// Slot method called when Preview
button is clicked.
   virtual void SetCurrentColor(Pixel_t col);/// Change current
color.
           void SetColorInfo(Int_t event, Int_t px, Int_t py, TO
bject *selected);
///  Set the color info in RGB and HLS parts
`
```

code

example

TGColorSelect

The TGColorFrame is a small frame with border showing a specific color.

The TG16ColorSelector is a composite frame with 16 TGColorFrames.

The TGColorPopup is a popup containing a TG16ColorSelector and a "More..." button which popups up a TGColorDialog allowing custom color selection.

The TGColorSelect widget is like a checkbutton but instead of the check mark there is color area with a little down arrow. When clicked on the arrow the TGColorPopup pops up.

```
Selecting a color in this widget will generate the event:

kC_COLORSEL, kCOL_SELCHANGED, widget id, pixel.

and the signal:

ColorSelected(Pixel_t pixel)
```

TGColorFrame 继承 TGFrame
Frame for color cell

TG16ColorSelector 继承 TGCompositeFrame
16 color cells

TGColorPopup 继承 TGCompositeFrame
Color selector popup

TGColorSelect 继承 TGCheckButton
Color selection checkbutton

class

TGColorFrame

```
   TGColorFrame(const TGWindow *p = 0, Pixel_t c = 0, Int_t n = 1
);
/// TGColorFrame constructor.
/// The TGColorFrame is a small frame with border showing a spec
ific color.

   virtual ~TGColorFrame() { }

   virtual Bool_t  HandleButton(Event_t *event);/// Handle butto
n events in TGColorFrame.
   virtual void    DrawBorder();/// Draw TGColorFrame border.

   void     SetActive(Bool_t in) { fActive = in; gClient->NeedRe
draw(this); }
   Pixel_t  GetColor() const { return fColor; }
```

TG16ColorSelector

```
   TG16ColorSelector(const TGWindow *p = 0);
/// TG16ColorSelector constructor.
/// The TG16ColorSelector is a composite frame with 16 TGColorFr
ames.

   virtual ~TG16ColorSelector();/// TG16ColorSelector destructor.


   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);/// Process messages for TG16ColorSelector.

   void    SetActive(Int_t newat);/// Set active color frame.
   Int_t   GetActive() { return fActive; }
```

TGColorPopup

```
   TGColorPopup(const TGWindow *p = 0, const TGWindow *m = 0, Pi
xel_t color = 0);
/// TGColorPopup constructor.
/// The TGColorPopup is a popup containing a TG16ColorSelector a
nd a "More..."
/// button which popups up a TGColorDialog allowing custom color
 selection.

   virtual ~TGColorPopup();/// TGColorPopup destructor.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button events for TGColorPopup.
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);/// Process messages for TGColorPopup.

   void    PlacePopup(Int_t x, Int_t y, UInt_t w, UInt_t h);///
Popup TGColorPopup at x,y position
   void    EndPopup();/// Ungrab pointer and unmap window.
   void    PreviewColor(Pixel_t color);/// Emit a signal to see
preview.
   void    PreviewAlphaColor(ULong_t color);/// Emit a signal to
 see preview.
```

TGColorSelect

```
   TGColorSelect(const TGWindow *p = 0, Pixel_t color = 0,
                 Int_t id = -1);
/// TGColorSelect constructor.
/// The TGColorSelect widget is like a checkbutton but instead o
f the check
/// mark there is color area with a little down arrow.
/// When clicked on the arrow the TGColorPopup pops up.

   virtual ~TGColorSelect();/// TGColorSelect destructor.

   virtual Bool_t HandleButton(Event_t *event);/// Handle button
 events for TGColorSelect.
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);/// Process messages for TGColorSelect.
```

```
   void    SetColor(Pixel_t color, Bool_t emit = kTRUE);/// Set
color.
   void    SetAlphaColor(ULong_t color, Bool_t emit = kTRUE);///
 Set color.
   Pixel_t GetColor() const { return fColor; }
   void    Enable(Bool_t on = kTRUE);  //*TOGGLE* *GETTER=IsEnab
led  /// Set state of widget as enabled.
   void    Disable();/// Set state of widget as disabled.

   // dummy methods just to remove from context menu
   void SetDown(Bool_t on = kTRUE, Bool_t emit = kFALSE) { TGBut
ton::SetDown(on, emit); }
   void Rename(const char *title)  { TGTextButton::SetTitle(titl
e); }
   void SetEnabled(Bool_t e = kTRUE) {TGButton::SetEnabled(e); }

   virtual TGDimension GetDefaultSize() const { return TGDimensi
on(43, 21); }
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
/// Save a color select widget as a C++ statement(s) on output s
tream out

   virtual void ColorSelected(Pixel_t color = 0)
            { Emit("ColorSelected(Pixel_t)", color ? color : Get
Color()); }  //*SIGNAL*
   virtual void AlphaColorSelected(ULong_t colptr = 0)
            { Emit("AlphaColorSelected(ULong_t)", colptr); }  //
*SIGNAL*
```

code

example

TGComboBox

TGComboBox 继承 TGCompositeFrame, TGWidget

A combobox (also known as a drop down listbox) allows the selection of one item out of a list of items. The selected item is visible in a little window. To view the list of possible items one has to click on a button on the right of the little window. This will drop down a listbox. After selecting an item from the listbox the box will disappear and the newly selected item will be shown in the little window.

```
Selecting an item in the combobox will generate the event:
kC_COMMAND, kCM_COMBOBOX, combobox id, item id.
```

TGLineStyleComboBox 继承 TGComboBox
The TGLineStyleComboBox user callable and it creates a combobox for selecting the line style.

TGLineWidthComboBox 继承 TGComboBox
The TGLineWidthComboBox user callable and it creates a combobox for selecting the line width.

TGFontTypeComboBox 继承 TGComboBox
The TGFontTypeComboBox is user callable and it creates a combobox for selecting the font.

class

TGComboBox

```
   TGComboBox(const TGWindow *p = 0, Int_t id = -1,
              UInt_t options = kHorizontalFrame | kSunkenFrame |
kDoubleBorder,
              Pixel_t back = GetWhitePixel());/// Create a combo
box widget.
```

```
   TGComboBox(const TGWindow *p, const char *text, Int_t id = -1
,
              UInt_t options = kHorizontalFrame | kSunkenFrame |
 kDoubleBorder,
              Pixel_t back = GetWhitePixel());/// Create an edit
able combo box widget.

   virtual ~TGComboBox();/// Delete a combo box widget.

   virtual void DrawBorder();/// Draw border of combo box widget.

   virtual TGDimension GetDefaultSize() const { return TGDimensi
on(fWidth, fHeight); }

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button events in the combo box.
   virtual Bool_t HandleDoubleClick(Event_t *event);/// Handle d
ouble click in text entry.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle pointe
r motion in text entry.
   virtual Bool_t HandleSelection(Event_t *event);/// Handle sel
ection in text entry.
   virtual Bool_t HandleSelectionRequest(Event_t *event);/// Han
dle selection request in text entry.
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);
/// Process messages generated by the listbox and forward
/// messages to the combobox message handling window. Parm2 cont
ains
/// the id of the selected listbox entry.

   virtual void AddEntry(TGString *s, Int_t id)
                        { fListBox->AddEntry(s, id); Resize(); }
   virtual void AddEntry(const char *s, Int_t id)
                        { fListBox->AddEntry(s, id); Resize(); }
   virtual void AddEntry(TGLBEntry *lbe, TGLayoutHints *lhints)
                        { fListBox->AddEntry(lbe, lhints); Resiz
e(); }
   virtual void InsertEntry(TGString *s, Int_t id, Int_t afterID)
```

```
                                { fListBox->InsertEntry(s, id, afterID);
 Resize(); }
   virtual void InsertEntry(const char *s, Int_t id, Int_t after
ID)
                                { fListBox->InsertEntry(s, id, afterID);
 Resize(); }
   virtual void InsertEntry(TGLBEntry *lbe, TGLayoutHints *lhint
s, Int_t afterID)
                                { fListBox->InsertEntry(lbe, lhints, aft
erID); Resize(); }
   virtual void NewEntry(const char *s = "Entry")
                                { fListBox->NewEntry(s); Resize(); }
   //*MENU*
   virtual void RemoveEntry(Int_t id = -1);
   //*MENU*
/// Remove entry. If id == -1, the currently selected entry is r
emoved

   virtual void RemoveAll();
   //*MENU*
/// Remove all entries from combo box.

   virtual void Layout();/// layout combobox
   virtual Bool_t IsTextInputEnabled() const { return (fTextEntr
y != 0); }
   virtual void EnableTextInput(Bool_t on);    //*TOGGLE* *GETTE
R=IsTextInputEnabled
/// Switch text input or readonly mode of combobox (not perfect
yet).

   virtual void RemoveEntries(Int_t from_ID, Int_t to_ID)
                                { fListBox->RemoveEntries(from_ID, to_ID
); }
   virtual Int_t GetNumberOfEntries() const
                                { return fListBox->GetNumberOfEntries();
 }

   virtual TGListBox    *GetListBox() const { return fListBox; }
   virtual TGTextEntry  *GetTextEntry() const { return fTextEntr
y; }
```

```
   virtual TGLBEntry    *FindEntry(const char *s) const;/// Find
 entry by name.
   virtual void  Select(Int_t id, Bool_t emit = kTRUE);/// Emit
signal.
/// Make the selected item visible in the combo box window
/// and emit signals according to the second parameter.

   virtual Int_t GetSelected() const { return fListBox->GetSelec
ted(); }
   virtual TGLBEntry *GetSelectedEntry() const
                        { return fListBox->GetSelectedEntry(); }
   virtual void SetTopEntry(TGLBEntry *e, TGLayoutHints *lh);
/// Set a new combo box value (normally update of text string in
/// fSelEntry is done via fSelEntry::Update()).

   virtual void SetEnabled(Bool_t on = kTRUE);   //*TOGGLE* *GET
TER=IsEnabled
/// Set state of combo box. If kTRUE=enabled, kFALSE=disabled.

   virtual Bool_t IsEnabled() const { return  fDDButton->IsEnabl
ed(); }
   virtual void SortByName(Bool_t ascend = kTRUE)
                  { fListBox->SortByName(ascend); }            /
/*MENU*icon=bld_sortup.png*

   virtual void Selected(Int_t widgetId, Int_t id);
     // *SIGNAL*
   virtual void Selected(Int_t id) { Emit("Selected(Int_t)", id)
; }  // *SIGNAL*
   virtual void Selected(const char *txt) { Emit("Selected(char*
)", txt); } // *SIGNAL*
   virtual void ReturnPressed();
     // *SIGNAL*
/// Add new entry to combo box when return key pressed inside te
xt entry
/// ReturnPressed signal is emitted.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a combo box widget as a C++ statement(s) on output stre
```

```
am out.
```

TGLineStyleComboBox

```
   TGLineStyleComboBox(const TGWindow *p = 0, Int_t id = -1,
               UInt_t options = kHorizontalFrame | kSunkenFrame |
 kDoubleBorder,
               Pixel_t back = GetWhitePixel());/// Create a line
style combo box.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a line style combo box widget as a C++ statement(s).
```

TGLineWidthComboBox

```
   TGLineWidthComboBox(const TGWindow *p = 0, Int_t id = -1,
               UInt_t options = kHorizontalFrame | kSunkenFrame |
 kDoubleBorder,
               Pixel_t back = GetWhitePixel(), Bool_t none=kFALSE
);
/// Create a line width combo box.
/// If "none" is equal to kTRUE the first entry is "None".

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a line width combo box widget as a C++ statement(s).
```

TGFontTypeComboBox

```
   TGFontTypeComboBox(const TGWindow *p = 0, Int_t id = -1,
             UInt_t options = kHorizontalFrame | kSunkenFrame | k
DoubleBorder,
             Pixel_t bask = GetWhitePixel());/// Create a text fo
nt combo box.
   virtual ~TGFontTypeComboBox();/// Text font combo box dtor.
```

code

```
#include "TGComboBox.h"

// TGComboBox
// 下拉选项
TGComboBox *fMonthBox;
fMonthBox = new TGComboBox(frame);
frame->AddFrame(fMonthBox, new TGLayoutHints(kLHintsLeft, 5, 5, 2
, 2));
fMonthBox->AddEntry("AAA", 1);
fMonthBox->AddEntry("BBB", 2);
fMonthBox->AddEntry("CCC", 3);
fMonthBox->Select(1);
// fMonthBox->Resize(100, /*fYearEntry*/->GetHeight());
```

example

TGDoubleSlider

```
//////////////////////////////////////////////////////////////////
//////////
//
        //
// TGDoubleSlider, TGDoubleVSlider and TGDoubleHSlider
        //
//
        //
// DoubleSlider widgets allow easy selection of a min and a max
value   //
// out of a range.
        //
// DoubleSliders can be either horizontal or vertical oriented a
nd      //
// there is a choice of three different types of tick marks.
        //
//
        //
// To change the min value press the mouse near to the left / bo
ttom    //
// edge of the slider.
        //
// To change the max value press the mouse near to the right / t
op      //
// edge of the slider.
        //
// To change both values simultaneously press the mouse near to
the     //
// center of the slider.
        //
//
        //
// TGDoubleSlider is an abstract base class. Use the concrete
        //
// TGDoubleVSlider and TGDoubleHSlider.
```

```
        //
//
        //
// Dragging the slider will generate the event:
        //
// kC_VSLIDER, kSL_POS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_POS, slider id, 0  (for horizontal slider)
        //
//
        //
// Pressing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)
        //
//
        //
// Releasing the mouse will generate the event:
        //
// kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)
        //
// kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider
)       //
//
        //
// Use the functions GetMinPosition(), GetMaxPosition() and
        //
// GetPosition() to retrieve the position of the slider.
        //
//
        //
//////////////////////////////////////////////////////////////////
//////////
```

TGDoubleSlider 继承 TGFrame, TGWidget

TGDoubleVSlider 继承 TGDoubleSlider

TGDoubleHSlider 继承 TGDoubleSlider

class

```
enum EDoubleSliderSize {
   //--- sizes for vert. and horz. sliders
   kDoubleSliderWidth  = 24,
   kDoubleSliderHeight = kDoubleSliderWidth
};


enum EDoubleSliderScale {
   //--- type of slider scale
   kDoubleScaleNo        = BIT(0),
   kDoubleScaleDownRight = BIT(1),
   kDoubleScaleBoth      = BIT(2)
};
```

TGDoubleSlider

```
   TGDoubleSlider(const TGWindow *p = 0, UInt_t w = 1, UInt_t h
= 1, UInt_t type = 1, Int_t id = -1,
                  UInt_t options = kChildFrame,
                  Pixel_t back = GetDefaultFrameBackground(),
                  Bool_t reversed = kFALSE,
                  Bool_t mark_ends = kFALSE);/// Slider construc
tor.

   virtual ~TGDoubleSlider() { }

   virtual Bool_t HandleButton(Event_t *event) = 0;
   virtual Bool_t HandleMotion(Event_t *event) = 0;

   virtual void  SetScale(Int_t scale) { fScale = scale; }
   virtual void  SetRange(Float_t min, Float_t max) {
      fVmin = min; fVmax = max;
      FixBounds(fVmin, fVmax);
   }
```

```
   virtual void SetPosition(Float_t min, Float_t max) {
      if (fReversedScale) { fSmin = fVmin+fVmax-max; fSmax = fVm
in+fVmax-min; }
      else { fSmin = min; fSmax = max; }
      fClient->NeedRedraw(this);
   }

   virtual Float_t GetMinPosition() const {
      if (fReversedScale) return fVmin+fVmax-fSmax;
      else return fSmin;
   }
   virtual Float_t GetMaxPosition() const {
      if (fReversedScale) return fVmin+fVmax-fSmin;
      else return fSmax;
   }
   virtual void GetPosition(Float_t &min, Float_t &max) const {
      if (fReversedScale) { min = fVmin+fVmax-fSmax; max = fVmin
+fVmax-fSmin; }
      else { min = fSmin; max = fSmax; }
   }
   virtual void GetPosition(Float_t *min, Float_t *max) const {
      if (fReversedScale) { *min = fVmin+fVmax-fSmax; *max = fVm
in+fVmax-fSmin; }
      else { *min = fSmin; *max = fSmax; }
   }

   virtual void  MapSubwindows() { TGWindow::MapSubwindows(); }

   virtual void  PositionChanged() { Emit("PositionChanged()");
} //*SIGNAL*
   virtual void  Pressed() { Emit("Pressed()"); }
  //*SIGNAL*
   virtual void  Released() { Emit("Released()"); }
  //*SIGNAL*
```

TGDoubleVSlider

```
   TGDoubleVSlider(const TGWindow *p = 0, UInt_t h = 1, UInt_t t
ype = 1, Int_t id = -1,
                    UInt_t options = kVerticalFrame,
                    Pixel_t back = GetDefaultFrameBackground(),
                    Bool_t reversed = kFALSE,
                    Bool_t mark_ends = kFALSE);/// Create a verti
cal slider widget.

   virtual ~TGDoubleVSlider();/// Delete vertical slider widget.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in vertical slider.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion event in vertical slider.
   virtual TGDimension GetDefaultSize() const
                      { return TGDimension(kDoubleSliderWidth, fH
eight); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save an horizontal slider as a C++ statement(s) on output st
ream out.
```

TGDoubleHSlider

```
   TGDoubleHSlider(const TGWindow *p = 0, UInt_t w = 1, UInt_t t
ype = 1, Int_t id = -1,
                    UInt_t options = kHorizontalFrame,
                    Pixel_t back = GetDefaultFrameBackground(),
                    Bool_t reversed = kFALSE,
                    Bool_t mark_ends = kFALSE);/// Create horizon
tal slider widget.

   virtual ~TGDoubleHSlider();/// Delete a horizontal slider wid
get.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in horizontal slider widget.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion event in horizontal slide widget.
   virtual TGDimension GetDefaultSize() const
                      { return TGDimension(fWidth, kDoubleSliderH
eight); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save an horizontal slider as a C++ statement(s) on output st
ream out.
```

code

example

TGFont

TGFont and TGFontPool

Encapsulate fonts used in the GUI system.
TGFontPool provides a pool of fonts. TGTextLayout is used to keep track of string measurement
information when using the text layout facilities.
It can be displayed with respect to any origin.

TGTextLayout 继承 TObject , friend TGFont

TGFont 继承 TNamed, TRefCnt ,friend TGFontPool, TGTextLayout;

TGFontPool 继承 TGObject

class

```
// Flags passed to TGFont::MeasureChars and TGFont::ComputeTextL
ayout

enum ETextLayoutFlags {
   kTextWholeWords = BIT(0),
   kTextAtLeastOne = BIT(1),
   kTextPartialOK  = BIT(2),
   kTextIgnoreTabs = BIT(3),
   kTextIgnoreNewlines = BIT(4)
};

enum EFontWeight {
   kFontWeightNormal = 0,
   kFontWeightMedium = 0,
   kFontWeightBold = 1,
   kFontWeightLight = 2,
   kFontWeightDemibold = 3,
   kFontWeightBlack = 4,
   kFontWeightUnknown = -1
```

```
};

enum EFontSlant {
   kFontSlantRoman = 0,
   kFontSlantItalic = 1,
   kFontSlantOblique = 2,
   kFontSlantUnknown = -1
};


struct FontMetrics_t {
   Int_t   fAscent;          // from baseline to top of font
   Int_t   fDescent;         // from baseline to bottom of font
   Int_t   fLinespace;       // the sum of the ascent and descent

   Int_t   fMaxWidth;        // width of widest character in font

   Bool_t  fFixed;           // true if monospace, false otherwi
se
};


struct FontAttributes_t {

   const char *fFamily; // Font family. The most important field.

   Int_t fPointsize;    // Pointsize of font, 0 for default size
, or negative number meaning pixel size.
   Int_t fWeight;       // Weight flag; see below for def'n.
   Int_t fSlant;        // Slant flag; see below for def'n.
   Int_t fUnderline;    // Non-zero for underline font.
   Int_t fOverstrike;   // Non-zero for overstrike font.

   FontAttributes_t():  // default constructor
      fFamily    (0),
      fPointsize (0),
      fWeight    (kFontWeightNormal),
      fSlant     (kFontSlantRoman),
      fUnderline (0),
      fOverstrike(0) { }
```

```
   FontAttributes_t(const FontAttributes_t& f): // copy construc
tor
      fFamily    (f.fFamily),
      fPointsize (f.fPointsize),
      fWeight    (f.fWeight),
      fSlant     (f.fSlant),
      fUnderline (f.fUnderline),
      fOverstrike(f.fOverstrike) { }

   FontAttributes_t& operator=(const FontAttributes_t& f) // ass
ignment operator
   {
      if (this != &f) {
         fFamily     = f.fFamily;
         fPointsize  = f.fPointsize;
         fWeight     = f.fWeight;
         fSlant      = f.fSlant;
         fUnderline  = f.fUnderline;
         fOverstrike = f.fOverstrike;
      }
      return *this;
   }

};
```

TGTextLayout

```
   TGTextLayout(): fFont(NULL), fString(""), fWidth(0), fNumChun
ks(0), fChunks(NULL) {}
   virtual ~TGTextLayout();/// destructor

   void   DrawText(Drawable_t dst, GContext_t gc, Int_t x, Int_t
 y,
                   Int_t firstChar, Int_t lastChar) const;
/// Use the information in the TGTextLayout object to display a
multi-line,
/// justified string of text.
/// This procedure is useful for simple widgets that need to dis
```

```
play
/// single-font, multi-line text and want TGFont to handle the d
etails.
/// dst       -- Window or pixmap in which to draw.
/// gc        -- Graphics context to use for drawing text.
/// x, y      -- Upper-left hand corner of rectangle in which to
 draw
///              (pixels).
/// firstChar -- The index of the first character to draw from t
he given
///              text item. 0 specfies the beginning.
/// lastChar  -- The index just after the last character to draw
 from the
///              given text item. A number < 0 means to draw all
 characters.

   void   UnderlineChar(Drawable_t dst, GContext_t gc,
                        Int_t x, Int_t y, Int_t underline) const
;
/// Use the information in the TGTextLayout object to display an
 underline
/// below an individual character. This procedure does not draw
the text,
/// just the underline.
/// This procedure is useful for simple widgets that need to dis
play
/// single-font, multi-line text with an individual character un
derlined
/// and want TGFont to handle the details. To display larger amo
unts of
/// underlined text, construct and use an underlined font.
/// dst       -- Window or pixmap in which to draw.
/// gc        -- Graphics context to use for drawing text.
/// x, y      -- Upper-left hand corner of rectangle in which to
 draw
///              (pixels).
/// underline -- Index of the single character to underline, or
-1 for
///              no underline.
```

```
   Int_t  PointToChar(Int_t x, Int_t y) const;
/// Use the information in the TGTextLayout token to determine t
he character
/// closest to the given point. The point must be specified with
 respect to
/// the upper-left hand corner of the text layout, which is cons
idered to be
/// located at (0, 0).
/// Any point whose y-value is less that 0 will be considered cl
osest to the
/// first character in the text layout; any point whose y-value
is greater
/// than the height of the text layout will be considered closes
t to the last
/// character in the text layout.
/// Any point whose x-value is less than 0 will be considered cl
osest to the
/// first character on that line; any point whose x-value is gre
ater than the
/// width of the text layout will be considered closest to the l
ast character
/// on that line.
/// The return value is the index of the character that was clos
est to the
/// point. Given a text layout with no characters, the value 0 w
ill always
/// be returned, referring to a hypothetical zero-width placehol
der character.

   Int_t  CharBbox(Int_t index, Int_t *x, Int_t *y, Int_t *w, In
t_t *h) const;
/// Use the information in the TGTextLayout token to return the
bounding box
/// for the character specified by index.
/// The width of the bounding box is the advance width of the ch
aracter, and
/// does not include and left- or right-bearing. Any character t
hat extends
/// partially outside of the text layout is considered to be tru
ncated at the
```

```
/// edge. Any character which is located completely outside of t
he text
/// layout is considered to be zero-width and pegged against the
 edge.
/// The height of the bounding box is the line height for this f
ont,
/// extending from the top of the ascent to the bottom of the de
scent.
/// Information about the actual height of the individual letter
 is not
/// available.
/// A text layout that contains no characters is considered to c
ontain a
/// single zero-width placeholder character.
/// The return value is 0 if the index did not specify a charact
er in the
/// text layout, or non-zero otherwise. In that case, *bbox is f
illed with
/// the bounding box of the character.
/// layout -- Layout information, from a previous call to Comput
eTextLayout().
/// index  -- The index of the character whose bbox is desired.
/// x, y   -- Filled with the upper-left hand corner, in pixels,
 of the
///           bounding box for the character specified by index,
 if non-NULL.
/// w, h   -- Filled with the width and height of the bounding b
ox for the
///           character specified by index, if non-NULL.

   Int_t  DistanceToText(Int_t x, Int_t y) const;
/// Computes the distance in pixels from the given point to the
given
/// text layout. Non-displaying space characters that occur at t
he end of
/// individual lines in the text layout are ignored for hit dete
ction
/// purposes.
/// The return value is 0 if the point (x, y) is inside the text
 layout.
```

```
/// If the point isn't inside the text layout then the return va
lue is the
/// distance in pixels from the point to the text item.
/// x, y -- Coordinates of point to check, with respect to the u
pper-left
///         corner of the text layout (in pixels).

   Int_t  IntersectText(Int_t x, Int_t y, Int_t w, Int_t h) const
;
/// Determines whether a text layout lies entirely inside, entir
ely outside,
/// or overlaps a given rectangle. Non-displaying space characte
rs that occur
/// at the end of individual lines in the text layout are ignore
d for
/// intersection calculations.
/// The return value is -1 if the text layout is entirely outsid
e of the
/// rectangle, 0 if it overlaps, and 1 if it is entirely inside
of the
/// rectangle.
/// x, y -- Upper-left hand corner, in pixels, of rectangular ar
ea to compare
///         with text layout. Coordinates are with respect to th
e upper-left
///         hand corner of the text layout itself.
/// w, h -- The width and height of the above rectangular area,
in pixels.

   void   ToPostscript(TString *dst) const;
/// Outputs the contents of a text layout in Postscript format.
The set of
/// lines in the text layout will be rendered by the user suppli
ed Postscript
/// function. The function should be of the form:
///     justify x y string  function  --
/// Justify is -1, 0, or 1, depending on whether the following s
tring should
/// be left, center, or right justified, x and y is the location
 for the
```

```
/// origin of the string, string is the sequence of characters t
o be printed,
/// and function is the name of the caller-provided function; th
e function
/// should leave nothing on the stack.
/// The meaning of the origin of the string (x and y) depends on
 the
/// justification. For left justification, x is where the left e
dge of the
/// string should appear. For center justification, x is where t
he center of
/// the string should appear. And for right justification, x is
where the
/// right edge of the string should appear. This behavior is nec
essary
/// because, for example, right justified text on the screen is
justified
/// with screen metrics. The same string needs to be justified w
ith printer
/// metrics on the printer to appear in the correct place with r
espect to
/// other similarly justified strings. In all circumstances, y i
s the
/// location of the baseline for the string.
/// result is modified to hold the Postscript code that will ren
der the text
/// layout.
```

TGFont

```
   virtual ~TGFont();/// Delete font.

   FontH_t      GetFontHandle() const { return fFontH; }
   FontStruct_t GetFontStruct() const { return fFontStruct; }
   FontStruct_t operator()() const;
   void         GetFontMetrics(FontMetrics_t *m) const;/// Get f
ont metrics.
   FontAttributes_t GetFontAttributes() const { return fFA; }
```

```
   Int_t  PostscriptFontName(TString *dst) const;
/// Return the name of the corresponding Postscript font for thi
s TGFont.
/// The return value is the pointsize of the TGFont. The name of
 the
/// Postscript font is appended to ds.
/// If the font does not exist on the printer, the print job wil
l fail at
/// print time. Given a "reasonable" Postscript printer, the fol
lowing
/// TGFont font families should print correctly:
///     Avant Garde, Arial, Bookman, Courier, Courier New, Genev
a,
///     Helvetica, Monaco, New Century Schoolbook, New York,
///     Palatino, Symbol, Times, Times New Roman, Zapf Chancery,
///     and Zapf Dingbats.
/// Any other TGFont font families may not print correctly becau
se the
/// computed Postscript font name may be incorrect.
/// dst -- Pointer to an initialized TString object to which the
 name of the
///        Postscript font that corresponds to the font will be
appended.

   Int_t  TextWidth(const char *string, Int_t numChars = -1) con
st;
/// A wrapper function for the more complicated interface of Mea
sureChars.
/// Computes how much space the given simple string needs.
/// The return value is the width (in pixels) of the given strin
g.
/// string   -- String whose width will be computed.
/// numChars -- Number of characters to consider from string, or
 < 0 for
///             strlen().

   Int_t  XTextWidth(const char *string, Int_t numChars = -1) co
nst;/// Return text widht in pixels
   Int_t  TextHeight() const { return fFM.fLinespace; }
   void   UnderlineChars(Drawable_t dst, GContext_t gc,
```

```
                          const char *string, Int_t x, Int_t y,
                          Int_t firstChar, Int_t lastChar) const;
/// This procedure draws an underline for a given range of chara
cters in a
/// given string. It doesn't draw the characters (which are assu
med to have
/// been displayed previously); it just draws the underline. Thi
s procedure
/// would mainly be used to quickly underline a few characters w
ithout having
/// to construct an underlined font. To produce properly underli
ned text, the
/// appropriate underlined font should be constructed and used.
/// dst       -- Window or pixmap in which to draw.
/// gc        -- Graphics context for actually drawing line.
/// string    -- String containing characters to be underlined o
r overstruck.
/// x, y      -- Coordinates at which first character of string
is drawn.
/// firstChar -- Index of first character.
/// lastChar  -- Index of one after the last character.

   TGTextLayout *ComputeTextLayout(const char *string, Int_t num
Chars,
                                   Int_t wrapLength, Int_t justif
y, Int_t flags,
                                   UInt_t *width, UInt_t *height)
const;
/// Computes the amount of screen space needed to display a mult
i-line,
/// justified string of text. Records all the measurements that
were done
/// to determine to size and positioning of the individual lines
 of text;
/// this information can be used by the TGTextLayout::DrawText()
 procedure
/// to display the text quickly (without remeasuring it).
/// This procedure is useful for simple widgets that want to dis
play
/// single-font, multi-line text and want TGFont to handle the d
```

```
etails.
/// The return value is a TGTextLayout token that holds the meas
urement
/// information for the given string. The token is only valid fo
r the given
/// string. If the string is freed, the token is no longer valid
 and must
/// also be deleted.
/// The dimensions of the screen area needed to display the text
 are stored
/// in *width and *height.
/// string     -- String whose dimensions are to be computed.
/// numChars   -- Number of characters to consider from string,
or < 0 for
///               strlen().
/// wrapLength -- Longest permissible line length, in pixels. <=
 0 means no
///               automatic wrapping: just let lines get as long
 as needed.
/// justify    -- How to justify lines.
/// flags      -- Flag bits OR-ed together. kTextIgnoreTabs mean
s that tab
///               characters should not be expanded. kTextIgnore
Newlines
///               means that newline characters should not cause
 a line break.
/// width      -- Filled with width of string.
/// height     -- Filled with height of string.

   Int_t  MeasureChars(const char *source, Int_t numChars, Int_t
 maxLength,
                       Int_t flags, Int_t *length) const;
/// Determine the number of characters from the string that will
 fit in the
/// given horizontal span. The measurement is done under the ass
umption that
/// DrawChars() will be used to actually display the characters.
/// The return value is the number of characters from source tha
t fit into
/// the span that extends from 0 to maxLength. *length is filled
```

```
 with the
/// x-coordinate of the right edge of the last character that di
d fit.
/// source    -- Characters to be displayed. Need not be '\0' te
rminated.
/// numChars  -- Maximum number of characters to consider from s
ource string.
/// maxLength -- If > 0, maxLength specifies the longest permiss
ible line
///              length; don't consider any character that would
 cross this
///              x-position. If <= 0, then line length is unboun
ded and the
///              flags argument is ignored.
/// flags     -- Various flag bits OR-ed together:
///              TEXT_PARTIAL_OK means include the last char whi
ch only
///              partially fit on this line.
///              TEXT_WHOLE_WORDS means stop on a word boundary,
 if possible.
///              TEXT_AT_LEAST_ONE means return at least one cha
racter even
///              if no characters fit.
/// *length   -- Filled with x-location just after the terminati
ng character.

   void   DrawCharsExp(Drawable_t dst, GContext_t gc, const char
 *source,
                       Int_t numChars, Int_t x, Int_t y) const;
/// Draw a string of characters on the screen. DrawCharsExp() ex
pands
/// control characters that occur in the string to \X or \xXX se
quences.
/// DrawChars() just draws the strings.
/// dst      -- Window or pixmap in which to draw.
/// gc       -- Graphics context for drawing characters.
/// source   -- Characters to be displayed. Need not be'\0' term
inated.
///             For DrawChars(), all meta-characters (tabs, cont
rol
```

```
///              characters, and newlines) should be stripped out
 of the
///              string that is passed to this function. If they
are not
///              stripped out, they will be displayed as regular
printing
///              characters.
/// numChars -- Number of characters in string.
/// x, y     -- Coordinates at which to place origin of string w
hen drawing.

   void   DrawChars(Drawable_t dst, GContext_t gc, const char *s
ource,
                    Int_t numChars, Int_t x, Int_t y) const;
/// Perform a quick sanity check to ensure we won't overflow the
 X
/// coordinate space.

   void  Print(Option_t *option="") const;/// Print font info.
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
/// Save the used font as a C++ statement(s) on output stream ou
t.
```

TGFontPool

```
   TGFontPool(TGClient *client);/// Create a font pool.
   virtual ~TGFontPool();/// Cleanup font pool.

   TGFont  *GetFont(const char *font, Bool_t fixedDefault = kTRU
E);
/// Get the specified font.
/// The font can be one of the following forms:
///        XLFD (see X documentation)
///        "Family [size [style] [style ...]]"
/// Returns 0 if error or no font can be found.
/// If fixedDefault is false the "fixed" font will not be substi
tuted
/// as fallback when the asked for font does not exist.
```

```
   TGFont  *GetFont(const TGFont *font);
/// Use font, i.e. increases ref count of specified font. Return
s 0
/// if font is not found.

   TGFont  *GetFont(FontStruct_t font);/// Use font, i.e. increa
ses ref count of specified font.
   TGFont  *GetFont(const char *family, Int_t ptsize, Int_t weig
ht, Int_t slant);
/// Returns font specified bay family, pixel/point size, weight
and slant
///  negative value of ptsize means size in pixels
///  positive value of ptsize means size in points

   void     FreeFont(const TGFont *font);/// Free font. If ref c
ount is 0 delete font.
   TGFont  *FindFont(FontStruct_t font) const;
/// Find font based on its font struct. Returns 0 if font is not
 found.

   TGFont  *FindFontByHandle(FontH_t font) const;
/// Find font based on its font handle. Returns 0 if font is not
 found.

   char   **GetAttributeInfo(const FontAttributes_t *fa);
/// Return information about the font attributes as an array of
strings.
/// An array of FONT_NUMFIELDS strings is returned holding the v
alue of the
/// font attributes in the following order:
/// family size weight slant underline overstrike

   void     FreeAttributeInfo(char **info);/// Free attributes i
nfo.
   char   **GetFontFamilies();
/// Return information about the font families that are availabl
e on the
/// current display.
/// An array of strings is returned holding a list of all the av
```

```
ailable font
/// families. The array is terminated with a NULL pointer.

   void     FreeFontFamilies(char **f);/// Delete an array of fa
milies allocated GetFontFamilies() method
   Bool_t   ParseFontName(const char *string, FontAttributes_t *
fa);
/// Converts a string into a set of font attributes that can be
used to
/// construct a font.
/// The string can be one of the following forms:
///        XLFD (see X documentation)
///        "Family [size [style] [style ...]]"
/// The return value is kFALSE if the object was syntactically
/// invalid. Otherwise, fills the font attribute buffer with the
 values
/// parsed from the string and returns kTRUE. The structure must
 already be
/// properly initialized.

   const char *NameOfFont(TGFont *font);/// Given a font, return
 a textual string identifying it.

   void     Print(Option_t *option="") const;/// List all fonts
in the pool.
```

code

```
TGFont *font = fpool->GetFont("helvetica", -9, kFontWeightNormal
, kFontSlantRoman);
font->Print();
```

example

TGFontDialog

TGFrame

The frame classes describe the different "dressed" GUI windows.

The TGFrame class is a subclasses of TGWindow, and is used as base class for some simple widgets (buttons, labels, etc.).
It provides:

- position & dimension fields
- an 'options' attribute (see constant above)
- a generic event handler
- a generic layout mechanism
- a generic border

The TGCompositeFrame class is the base class for composite widgets (menu bars, list boxes, etc.).
It provides:

- a layout manager
- a frame container (TList *)

The TGVerticalFrame and TGHorizontalFrame are composite frame that layout their children in vertical or horizontal way.

The TGMainFrame class defines top level windows that interact with the system Window Manager.

The TGTransientFrame class defines transient windows that typically are used for dialogs windows.

The TGGroupFrame is a composite frame with a border and a title.
It is typically used to group a number of logically related widgets
visually together.

TGFrame 继承 TGWindow, TQObject
This class subclasses TGWindow, used as base class for some simple widgets (buttons, labels, etc.).
It provides:

- position & dimension fields
- an 'options' attribute (see constant above)
- a generic event handler
- a generic layout mechanism
- a generic border

TGCompositeFrame 继承 TGFrame
This class is the base class for composite widgets (menu bars, list boxes, etc.).
It provides:

- a layout manager
- a frame container (TList *)

TGVerticalFrame 继承 TGCompositeFrame
Composite frame with vertical child layout

TGHorizontalFrame 继承 TGCompositeFrame
Composite frame with horizontal child layout

TGMainFrame 继承 TGCompositeFrame
This class defines top level windows that interact with the system Window Manager (WM or MWM for Motif Window Manager).

TGTransientFrame 继承 TGMainFrame
This class defines transient windows that typically are used for dialogs.

TGGroupFrame 继承 TGCompositeFrame
A group frame is a composite frame with a border and a title. It is typically used to group a number of logically related widgets visually together.

TGHeaderFrame 继承 TGHorizontalFrame
Horizontal Frame used to contain header buttons and splitters in a list view. Used to have resizable column headers.

class

```
//---- frame states

enum EFrameState {
   kIsVisible  = BIT(0),
   kIsMapped   = kIsVisible,
   kIsArranged = BIT(1)
};

//---- frame cleanup
enum EFrameCleanup {
   kNoCleanup    = 0,
   kLocalCleanup = 1,
   kDeepCleanup  = -1
};

//---- types of frames (and borders)

enum EFrameType {
   kChildFrame      = 0,
   kMainFrame       = BIT(0),
   kVerticalFrame   = BIT(1),
   kHorizontalFrame = BIT(2),
   kSunkenFrame     = BIT(3),
   kRaisedFrame     = BIT(4),
   kDoubleBorder    = BIT(5),
   kFitWidth        = BIT(6),
   kFixedWidth      = BIT(7),
   kFitHeight       = BIT(8),
   kFixedHeight     = BIT(9),
   kFixedSize       = (kFixedWidth | kFixedHeight),
   kOwnBackground   = BIT(10),
   kTransientFrame  = BIT(11),
```

```
    kTempFrame        = BIT(12),
    kMdiMainFrame     = BIT(13),
    kMdiFrame         = BIT(14)
};

//---- MWM hints stuff

enum EMWMHints {
    // functions
    kMWMFuncAll       = BIT(0),
    kMWMFuncResize    = BIT(1),
    kMWMFuncMove      = BIT(2),
    kMWMFuncMinimize = BIT(3),
    kMWMFuncMaximize = BIT(4),
    kMWMFuncClose     = BIT(5),

    // input mode
    kMWMInputModeless                = 0,
    kMWMInputPrimaryApplicationModal = 1,
    kMWMInputSystemModal             = 2,
    kMWMInputFullApplicationModal    = 3,

    // decorations
    kMWMDecorAll       = BIT(0),
    kMWMDecorBorder    = BIT(1),
    kMWMDecorResizeH   = BIT(2),
    kMWMDecorTitle     = BIT(3),
    kMWMDecorMenu      = BIT(4),
    kMWMDecorMinimize = BIT(5),
    kMWMDecorMaximize = BIT(6)
};

//---- drag and drop

enum EDNDFlags {
    kIsDNDSource = BIT(0),
    kIsDNDTarget = BIT(1)
};
```

TGFrame

```
   // Default colors and graphics contexts
   static Pixel_t     GetDefaultFrameBackground();/// Get defaul
t frame background.
   static Pixel_t     GetDefaultSelectedBackground();/// Get def
ault selected frame background.
   static Pixel_t     GetWhitePixel();/// Get white pixel value.
   static Pixel_t     GetBlackPixel();/// Get black pixel value.
   static const TGGC &GetBlackGC();/// Get black graphics contex
t.
   static const TGGC &GetWhiteGC();/// Get white graphics contex
t.
   static const TGGC &GetHilightGC();/// Get highlight color gra
phics context.
   static const TGGC &GetShadowGC();/// Get shadow color graphic
s context.
   static const TGGC &GetBckgndGC();/// Get background color gra
phics context.

   TGFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
           UInt_t options = 0, Pixel_t back = GetDefaultFrameBac
kground());
/// Create a TGFrame object. Options is an OR of the EFrameTypes.


   TGFrame(TGClient *c, Window_t id, const TGWindow *parent = 0)
;
/// Create a frame using an externally created window. For examp
le
/// to register the root window (called by TGClient), or a window

/// created via TVirtualX::InitWindow() (id is obtained with
/// TVirtualX::GetWindowID()).

   virtual ~TGFrame();

   virtual void DeleteWindow();
/// Delete window. Use single shot timer to call final delete me
thod.
/// We use this indirect way since deleting the window in its own
```

```
/// execution "thread" can cause side effects because frame meth
ods
/// can still be called while the window object has already been
 deleted.

   virtual void ReallyDelete() { delete this; }

   UInt_t GetEventMask() const { return fEventMask; }
   void   AddInput(UInt_t emask);
/// Add events specified in the emask to the events the frame sh
ould handle.

   void   RemoveInput(UInt_t emask);
/// Remove events specified in emask from the events the frame s
hould handle.

   virtual Bool_t HandleEvent(Event_t *event);
/// Handle all frame events. Events are dispatched to the specif
ic
/// event handlers.
   virtual Bool_t HandleConfigureNotify(Event_t *event);
/// This event is generated when the frame is resized.

   virtual Bool_t HandleButton(Event_t *) { return kFALSE; }
   virtual Bool_t HandleDoubleClick(Event_t *) { return kFALSE;
}
   virtual Bool_t HandleCrossing(Event_t *) { return kFALSE; }
   virtual Bool_t HandleMotion(Event_t *) { return kFALSE; }
   virtual Bool_t HandleKey(Event_t *) { return kFALSE; }
   virtual Bool_t HandleFocusChange(Event_t *) { return kFALSE;
}
   virtual Bool_t HandleClientMessage(Event_t *event);
/// Handle a client message. Client messages are the ones sent v
ia
/// TGFrame::SendMessage (typically by widgets).

   virtual Bool_t HandleSelection(Event_t *) { return kFALSE; }
   virtual Bool_t HandleSelectionRequest(Event_t *) { return kFA
LSE; }
```

```
   virtual Bool_t HandleSelectionClear(Event_t *) { return kFALS
E; }
   virtual Bool_t HandleColormapChange(Event_t *) { return kFALS
E; }
   virtual Bool_t HandleDragEnter(TGFrame *) { return kFALSE; }
   virtual Bool_t HandleDragLeave(TGFrame *) { return kFALSE; }
   virtual Bool_t HandleDragMotion(TGFrame *) { return kFALSE; }
   virtual Bool_t HandleDragDrop(TGFrame *, Int_t /*x*/, Int_t /
*y*/, TGLayoutHints*)
                     { return kFALSE; }
   virtual void   ProcessedConfigure(Event_t *event)
                     { Emit("ProcessedConfigure(Event_t*)", (Lon
g_t)event); } //*SIGNAL*
   virtual void   ProcessedEvent(Event_t *event)
                     { Emit("ProcessedEvent(Event_t*)", (Long_t)
event); } //*SIGNAL*

   virtual void   SendMessage(const TGWindow *w, Long_t msg, Lon
g_t parm1, Long_t parm2);
/// Send message (i.e. event) to window w. Message is encoded in
 one long
/// as message type and up to two long parameters.

   virtual Bool_t ProcessMessage(Long_t, Long_t, Long_t) { return
 kFALSE; }

   virtual TGDimension GetDefaultSize() const ;
   virtual void    Move(Int_t x, Int_t y); /// Move frame.
   virtual void    Resize(UInt_t w = 0, UInt_t h = 0);
/// Resize the frame.
/// If w=0 && h=0 - Resize to default size

   virtual void    Resize(TGDimension size);/// Resize the frame.

   virtual void    MoveResize(Int_t x, Int_t y, UInt_t w = 0, UI
nt_t h = 0);
/// Move and/or resize the frame.
/// If w=0 && h=0 - Resize to default size

   virtual UInt_t  GetDefaultWidth() const { return GetDefaultSi
```

```
ze().fWidth; }
   virtual UInt_t  GetDefaultHeight() const { return GetDefaultS
ize().fHeight; }
   virtual Pixel_t GetBackground() const { return fBackground; }
   virtual void    ChangeBackground(Pixel_t back); /// Change fr
ame background color.
   virtual void    SetBackgroundColor(Pixel_t back);
/// Set background color (override from TGWindow base class).
/// Same effect as ChangeBackground().

   virtual Pixel_t GetForeground() const; /// Return frame foreg
round color.
   virtual void    SetForegroundColor(Pixel_t /*fore*/) { }
   virtual UInt_t  GetOptions() const { return fOptions; }
   virtual void    ChangeOptions(UInt_t options); /// Change fra
me options. Options is an OR of the EFrameTypes.
   virtual void    Layout() { }
   virtual void    MapSubwindows() { }  // Simple frames do not
have subwindows
                                        // Redefine this in TGCo
mpositeFrame!
   virtual void    ReparentWindow(const TGWindow *p, Int_t x = 0
, Int_t y = 0)
                     { TGWindow::ReparentWindow(p, x, y); Move(x
, y); }
   virtual void    MapWindow() { TGWindow::MapWindow(); if (fFE)
 fFE->fState |= kIsVisible; }
   virtual void    MapRaised() { TGWindow::MapRaised(); if (fFE)
 fFE->fState |= kIsVisible; }
   virtual void    UnmapWindow() { TGWindow::UnmapWindow(); if (
fFE) fFE->fState &= ~kIsVisible; }

   virtual void    DrawBorder(); /// Draw frame border.
   virtual void    DrawCopy(Handle_t /*id*/, Int_t /*x*/, Int_t
/*y*/) { }
   virtual void    Activate(Bool_t) { }
   virtual Bool_t  IsActive() const { return kFALSE; }
   virtual Bool_t  IsComposite() const { return kFALSE; }
   virtual Bool_t  IsEditable() const { return kFALSE; }
   virtual void    SetEditable(Bool_t) {}
```

```
   virtual void    SetLayoutBroken(Bool_t = kTRUE) {}
   virtual Bool_t  IsLayoutBroken() const { return kFALSE; }
   virtual void    SetCleanup(Int_t = kLocalCleanup) { /* backwa
rd compatebility */ }

   virtual void    SetDragType(Int_t type);
   virtual void    SetDropType(Int_t type);
   virtual Int_t   GetDragType() const;
/// Returns drag source type.
/// If frame is not "draggable" - return zero

   virtual Int_t   GetDropType() const;
/// Returns drop target type.
/// If frame cannot accept drop - return zero

   UInt_t GetWidth() const { return fWidth; }
   UInt_t GetHeight() const { return fHeight; }
   UInt_t GetMinWidth() const { return fMinWidth; }
   UInt_t GetMinHeight() const { return fMinHeight; }
   UInt_t GetMaxWidth() const { return fMaxWidth; }
   UInt_t GetMaxHeight() const { return fMaxHeight; }
   TGDimension GetSize() const { return TGDimension(fWidth, fHei
ght); }
   Int_t  GetX() const { return fX; }
   Int_t  GetY() const { return fY; }
   Int_t  GetBorderWidth() const { return fBorderWidth; }

   TGFrameElement *GetFrameElement() const { return fFE; }
   void SetFrameElement(TGFrameElement *fe) { fFE = fe; }

   Bool_t Contains(Int_t x, Int_t y) const
      { return ((x >= 0) && (x < (Int_t)fWidth) && (y >= 0) && (
y < (Int_t)fHeight)); }
   virtual TGFrame *GetFrameFromPoint(Int_t x, Int_t y)
      { return (Contains(x, y) ? this : 0); }

   // Modifiers (without graphic update)
   virtual void SetX(Int_t x) { fX = x; }
   virtual void SetY(Int_t y) { fY = y; }
   virtual void SetWidth(UInt_t w) { fWidth = w; }
```

```
   virtual void SetHeight(UInt_t h) { fHeight = h; }
   virtual void SetMinWidth(UInt_t w) { fMinWidth = w; }
   virtual void SetMinHeight(UInt_t h) { fMinHeight = h; }
   virtual void SetMaxWidth(UInt_t w) { fMaxWidth = w; }
   virtual void SetMaxHeight(UInt_t h) { fMaxHeight = h; }
   virtual void SetSize(const TGDimension &s) { fWidth = s.fWidt
h; fHeight = s.fHeight; }

   // Printing and saving
   virtual void Print(Option_t *option="") const;/// Print windo
w id.
   void SaveUserColor(std::ostream &out, Option_t *);/// Save a
user color in a C++ macro file - used in SavePrimitive().
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a frame widget as a C++ statement(s) on output stream o
ut.

   // dummy to remove from context menu
   virtual void        Delete(Option_t * /*option*/ ="") { }
   virtual TObject    *DrawClone(Option_t * /*option */="") const
 { return 0; }
   virtual void        DrawClass() const { }
   virtual void        Dump() const { }
   virtual void        Inspect() const { }
   virtual void        SetDrawOption(Option_t * /*option*/="") {
 }

   // drag and drop...
   void                SetDNDSource(Bool_t onoff)
                       { if (onoff) fDNDState |= kIsDNDSource; e
lse fDNDState &= ~kIsDNDSource; }
   void                SetDNDTarget(Bool_t onoff)
                       { if (onoff) fDNDState |= kIsDNDTarget; e
lse fDNDState &= ~kIsDNDTarget; }
   Bool_t              IsDNDSource() const { return fDNDState &
kIsDNDSource; }
   Bool_t              IsDNDTarget() const { return fDNDState &
kIsDNDTarget; }
```

```
   virtual TDNDData   *GetDNDData(Atom_t /*dataType*/) { return 0
; }
   virtual Bool_t      HandleDNDDrop(TDNDData * /*DNDData*/) { r
eturn kFALSE; }
   virtual Atom_t      HandleDNDPosition(Int_t /*x*/, Int_t /*y*/
, Atom_t /*action*/,
                                         Int_t /*xroot*/, Int_t
/*yroot*/) { return kNone; }
   virtual Atom_t      HandleDNDEnter(Atom_t * /*typelist*/) { r
eturn kNone; }
   virtual Bool_t      HandleDNDLeave() { return kFALSE; }
   virtual Bool_t      HandleDNDFinished() { return kFALSE; }
```

TGCompositeFrame

```
   TGCompositeFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t
h = 1,
                    UInt_t options = 0,
                    Pixel_t back = GetDefaultFrameBackground());
/// Create a composite frame. A composite frame has in addition
to a TGFrame
/// also a layout manager and a list of child frames.

   TGCompositeFrame(TGClient *c, Window_t id, const TGWindow *pa
rent = 0);
/// Create a frame using an externally created window. For examp
le
/// to register the root window (called by TGClient), or a window

/// created via TVirtualX::InitWindow() (id is obtained with TVi
rtualX::GetWindowID()).


   virtual ~TGCompositeFrame();/// Delete a composite frame.

   virtual TList *GetList() const { return fList; }

   virtual UInt_t GetDefaultWidth() const
                    { return GetDefaultSize().fWidth; }
```

```
   virtual UInt_t GetDefaultHeight() const
                     { return GetDefaultSize().fHeight; }
   virtual TGDimension GetDefaultSize() const
                     { return (IsLayoutBroken() ? TGDimension(fW
idth, fHeight) :
                               fLayoutManager->GetDefaultSize())
; }
   virtual TGFrame *GetFrameFromPoint(Int_t x, Int_t y);/// Get
frame located at specified point.
   virtual Bool_t TranslateCoordinates(TGFrame *child, Int_t x,
Int_t y,
                                       Int_t &fx, Int_t &fy);
/// Translate coordinates to child frame.

   virtual void   MapSubwindows();
/// Map all sub windows that are part of the composite frame.

   virtual void   Layout();/// Layout the elements of the compos
ite frame.
   virtual Bool_t HandleButton(Event_t *) { return kFALSE; }
   virtual Bool_t HandleDoubleClick(Event_t *) { return kFALSE;
}
   virtual Bool_t HandleCrossing(Event_t *) { return kFALSE; }
   virtual Bool_t HandleMotion(Event_t *) { return kFALSE; }
   virtual Bool_t HandleKey(Event_t *) { return kFALSE; }
   virtual Bool_t HandleFocusChange(Event_t *) { return kFALSE;
}
   virtual Bool_t HandleSelection(Event_t *) { return kFALSE; }
   virtual Bool_t HandleDragEnter(TGFrame *);/// Handle drag ent
er event.
   virtual Bool_t HandleDragLeave(TGFrame *);/// Handle drag lea
ve event.
   virtual Bool_t HandleDragMotion(TGFrame *);/// Handle drag mo
tion event.
   virtual Bool_t HandleDragDrop(TGFrame *frame, Int_t x, Int_t
y, TGLayoutHints *lo);/// Handle drop event.
   virtual void   ChangeOptions(UInt_t options);
/// Change composite frame options. Options is an OR of the EFra
meTypes.
```

```
   virtual Bool_t ProcessMessage(Long_t, Long_t, Long_t) { return
 kFALSE; }

   virtual TGLayoutManager *GetLayoutManager() const { return fL
ayoutManager; }
   virtual void SetLayoutManager(TGLayoutManager *l);
/// Set the layout manager for the composite frame.
/// The layout manager is adopted by the frame and will be delet
ed
/// by the frame.

   virtual TGFrameElement* FindFrameElement(TGFrame *f) const;//
/ Find frame-element holding frame f.

   virtual void   AddFrame(TGFrame *f, TGLayoutHints *l = 0);
/// Add frame to the composite frame using the specified layout
hints.
/// If no hints are specified default hints TGLayoutHints(kLHint
sNormal,0,0,0,0)
/// will be used. Most of the time, however, you will want to pr
ovide
/// specific hints. User specified hints can be reused many times

/// and need to be destroyed by the user. The added frames canno
t not be
/// added to different composite frames but still need to be del
eted by
/// the user.

   virtual void   RemoveAll(); /// Remove all frames from compos
ite frame.
   virtual void   RemoveFrame(TGFrame *f); /// Remove frame from
 composite frame.
   virtual void   ShowFrame(TGFrame *f);/// Show sub frame.
   virtual void   HideFrame(TGFrame *f);/// Hide sub frame.
   Int_t          GetState(TGFrame *f) const;/// Get state of su
b frame.
   Bool_t         IsVisible(TGFrame *f) const;/// Get state of s
ub frame.
   Bool_t         IsVisible(TGFrameElement *ptr) const { return
```

```
(ptr->fState & kIsVisible); }
   Bool_t         IsArranged(TGFrame *f) const;/// Get state of
sub frame.
   Bool_t         IsArranged(TGFrameElement *ptr) const { return
 (ptr->fState & kIsArranged); }
   Bool_t         IsComposite() const { return kTRUE; }
   virtual Bool_t IsEditable() const; /// Return kTRUE if frame
is being edited.
   virtual void   SetEditable(Bool_t on = kTRUE);
/// Switch ON/OFF edit mode.
/// If edit mode is ON it is possible:
///  1. embed other ROOT GUI application (a la ActiveX)

   virtual void   SetLayoutBroken(Bool_t on = kTRUE); /// Set br
oken layout. No Layout method is called.
   virtual Bool_t IsLayoutBroken() const
                  { return fLayoutBroken || !fLayoutManager; }
   virtual void   SetEditDisabled(UInt_t on = 1);
/// Set edit disable flag for this frame and subframes
///  - if (on & kEditDisable) - disable edit for this frame and
all subframes.

   virtual void   SetCleanup(Int_t mode = kLocalCleanup);
/// Turn on automatic cleanup of child frames in dtor.
/// if mode = kNoCleanup    - no automatic cleanup
/// if mode = kLocalCleanup - automatic cleanup in this composit
e frame only
/// if mode = kDeepCleanup  - automatic deep cleanup in this com
posite frame
///                           and all child composite frames (hi
erarchical)
/// Attention!
///    Hierarchical cleaning is dangerous and must be used with
caution.
///    There are many GUI components (in ROOT and in user code)
which do not
///    use Clean method in destructor ("custom deallocation").
///    Adding such component to GUI container which is using hie
rarchical
///    cleaning will produce seg. violation when container is de
```

```
leted.
///    The reason is double deletion: first whem Clean method is
 invoked,
///    then at "custom deallocation".
///    We are going to correct all ROOT code to make it to be
///    consitent with hierarchical cleaning scheeme.

   virtual Int_t  MustCleanup() const { return fMustCleanup; }
   virtual void   Cleanup();
/// Cleanup and delete all objects contained in this composite f
rame.
/// This will delete all objects added via AddFrame().
/// CAUTION: all objects (frames and layout hints) must be uniqu
e, i.e.
/// cannot be shared.

   virtual void   SetMapSubwindows(Bool_t on) {  fMapSubwindows
= on; }
   virtual Bool_t IsMapSubwindows() const { return fMapSubwindow
s; }

   virtual void   Print(Option_t *option="") const;/// Print all
 frames in this composite frame.
   virtual void   ChangeSubframesBackground(Pixel_t back);
/// Change background color for this frame and all subframes.

   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a composite frame widget as a C++ statement(s) on outpu
t stream out.

   virtual void   SavePrimitiveSubframes(std::ostream &out, Opti
on_t *option = "");
/// Auxilary protected method  used to save subframes.
```

TGVerticalFrame

```
   TGVerticalFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t h
 = 1,
                   UInt_t options = kChildFrame,
                   Pixel_t back = GetDefaultFrameBackground()) :
      TGCompositeFrame(p, w, h, options | kVerticalFrame, back)
{ SetWindowName(); }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a vertical frame widget as a C++ statement(s) on output
 stream out.
```

TGHorizontalFrame

```
   TGHorizontalFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t
 h = 1,
                     UInt_t options = kChildFrame,
                     Pixel_t back = GetDefaultFrameBackground()
 :
      TGCompositeFrame(p, w, h, options | kHorizontalFrame, back
) { SetWindowName(); }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a horizontal frame widget as a C++ statement(s) on outp
ut stream out.
```

TGMainFrame

```
   TGMainFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1
,
               UInt_t options = kVerticalFrame);
/// Create a top level main frame. A main frame interacts
/// with the window manager.

   virtual ~TGMainFrame();/// TGMainFrame destructor.

   virtual Bool_t HandleKey(Event_t *event);/// Handle keyboard
events.
   virtual Bool_t HandleClientMessage(Event_t *event);/// Handle
```

```
 client messages sent to this frame.
   virtual Bool_t HandleSelection(Event_t *event);/// Handle pri
mary selection event.
   virtual Bool_t HandleSelectionRequest(Event_t *event);/// Han
dle selection request event.
   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button events.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion events.
   virtual Bool_t SaveFrameAsCodeOrImage();
/// Opens dialog window allowing user to save the frame contents
/// as a ROOT macro or as an image.
/// Returns kTRUE if something was saved.
/// This is bound to Ctrl-S by default.

   virtual void   SendCloseMessage();
/// Send close message to self. This method should be called from

/// a button to close this window.

   virtual void   CloseWindow();   //*SIGNAL*
/// Close and delete main frame. We get here in response to ALT+
F4 or
/// a window manager close command. To terminate the application
 when this
/// happens override this method and call gApplication->Terminat
e(0) or
/// make a connection to this signal (if after the slot this met
hod
/// should not be called call DontCallClose() in the slot).
/// By default the window will be deleted.

   void DontCallClose();
/// Typically call this method in the slot connected to the Clos
eWindow()
/// signal to prevent the calling of the default or any derived
CloseWindow()
/// methods to prevent premature or double deletion of this wind
ow.
```

```
   void SetWindowName(const char *name = 0);
/// Set window name. This is typically done via the window manag
er.

   void SetIconName(const char *name);
/// Set window icon name. This is typically done via the window
manager.

   const TGPicture *SetIconPixmap(const char *iconName);
/// Set window icon pixmap by name. This is typically done via t
he window
/// manager. Icon can be in any image format supported by TImage
, e.g.
/// GIF, XPM, PNG, JPG .. or even PS, PDF (see EImageFileTypes i
n TImage.h
/// for the full list of supported formats).

   void SetIconPixmap(char **xpm_array);
/// Set window icon by xpm array. That allows to have icons
/// builtin to the source code.

   void SetClassHints(const char *className, const char *resourc
eName);
/// Set the windows class and resource name. Used to get the rig
ht
/// resources from the resource database. However, ROOT applicat
ions
/// will typically use the .rootrc file for this.

   void SetMWMHints(UInt_t value, UInt_t funcs, UInt_t input);
/// Set decoration style for MWM-compatible wm (mwm, ncdwm, fvwm
?).

   void SetWMPosition(Int_t x, Int_t y);/// Give the window mana
ger a window position hint.
   void SetWMSize(UInt_t w, UInt_t h);/// Give the window manage
r a window size hint.
   void SetWMSizeHints(UInt_t wmin, UInt_t hmin, UInt_t wmax, UI
nt_t hmax,
                       UInt_t winc, UInt_t hinc);
```

```
/// Give the window manager minimum and maximum size hints. Also
/// specify via winc and hinc the resize increments.

   void SetWMState(EInitialState state);/// Set the initial stat
e of the window. Either kNormalState or kIconicState.

   virtual Bool_t BindKey(const TGWindow *w, Int_t keycode, Int_
t modifier) const;/// Bind key to a window.
   virtual void   RemoveBind(const TGWindow *w, Int_t keycode, I
nt_t modifier) const;/// Remove key binding.
   TList *GetBindList() const { return fBindList; }

   const char *GetWindowName() const { return fWindowName; }
   const char *GetIconName() const { return fIconName; }
   const char *GetIconPixmap() const { return fIconPixmap; }
   void GetClassHints(const char *&className, const char *&resou
rceName) const
      { className = fClassName.Data(); resourceName = fResourceN
ame.Data(); }
   void GetMWMHints(UInt_t &value, UInt_t &funcs, UInt_t &input)
const
      { value = fMWMValue; funcs = fMWMFuncs; input = fMWMInput;
 }
   void GetWMPosition(Int_t &x, Int_t &y) const { x = fWMX; y =
fWMY; }
   void GetWMSize(UInt_t &w, UInt_t &h) const { w = fWMWidth; h
= fWMHeight; }
   void GetWMSizeHints(UInt_t &wmin, UInt_t &hmin, UInt_t &wmax,
 UInt_t &hmax,
                       UInt_t &winc, UInt_t &hinc) const
      { wmin = fWMMinWidth; hmin = fWMMinHeight; wmax = fWMMaxWi
dth;
        hmax = fWMMaxHeight; winc = fWMWidthInc; hinc = fWMHeigh
tInc; }
   EInitialState GetWMState() const { return fWMInitState; }

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a main frame widget as a C++ statement(s) on output str
eam out.
```

```
   virtual void SaveSource(const char *filename = "Rootappl.C",
Option_t *option = ""); // *MENU*icon=bld_save.png*
/// Save the GUI main frame widget in a C++ macro file.
```

TGTransientFrame

```
   TGTransientFrame(const TGWindow *p = 0, const TGWindow *main
= 0, UInt_t w = 1, UInt_t h = 1,
                    UInt_t options = kVerticalFrame);
/// Create a transient window. A transient window is typically u
sed for
/// dialog boxes.

   enum EPlacement { kCenter, kLeft, kRight, kTop, kBottom, kTop
Left, kTopRight,
                     kBottomLeft, kBottomRight };
   virtual void    CenterOnParent(Bool_t croot = kTRUE, EPlaceme
nt pos = kCenter);
/// Position transient frame centered relative to the parent fra
me.
/// If fMain is 0 (i.e. TGTransientFrame is acting just like a
/// TGMainFrame) and croot is true, the window will be centered
on
/// the root window, otherwise no action is taken and the default

/// wm placement will be used.

   const TGWindow *GetMain() const { return fMain; }
   virtual void    SavePrimitive(std::ostream &out, Option_t *op
tion = "");
/// Save a transient frame widget as a C++ statement(s) on outpu
t stream out.

   virtual void    SaveSource(const char *filename = "Rootdlog.C"
, Option_t *option = ""); // *MENU*icon=bld_save.png*
/// Save the GUI tranzient frame widget in a C++ macro file.
```

TGGroupFrame

```
   enum ETitlePos { kLeft = -1, kCenter = 0, kRight = 1 };

   static FontStruct_t  GetDefaultFontStruct();/// Return defaul
t font structure in use.
   static const TGGC   &GetDefaultGC();/// Return default graphi
cs context in use.

   TGGroupFrame(const TGWindow *p, TGString *title,
                UInt_t options = kVerticalFrame,
                GContext_t norm = GetDefaultGC()(),
                FontStruct_t font = GetDefaultFontStruct(),
                Pixel_t back = GetDefaultFrameBackground());
/// Create a group frame. The title will be adopted and deleted
by the
/// group frame.

   TGGroupFrame(const TGWindow *p = 0, const char *title = 0,
                UInt_t options = kVerticalFrame,
                GContext_t norm = GetDefaultGC()(),
                FontStruct_t font = GetDefaultFontStruct(),
                Pixel_t back = GetDefaultFrameBackground());
/// Create a group frame.

   virtual ~TGGroupFrame();/// Delete a group frame.

   virtual TGDimension GetDefaultSize() const;/// Returns defaul
t size.
   virtual void  DrawBorder();
/// Draw border of around the group frame.
/// if frame is kRaisedFrame  - a frame border is of "wall style
",
/// otherwise of "groove style".

   virtual void  SetTitle(TGString *title);
/// Set or change title of the group frame. Titlte TGString is a
dopted
/// by the TGGroupFrame.
```

```
   virtual void  SetTitle(const char *title);/// Set or change t
itle of the group frame.

   virtual void  Rename(const char *title)  { SetTitle(title); }
//*MENU*icon=bld_rename.png*
           Int_t GetTitlePos() const { return fTitlePos; }
   virtual void  SetTitlePos(ETitlePos pos = kLeft) { fTitlePos
= pos; }  //*SUBMENU*
   virtual void  SetTextColor(Pixel_t color, Bool_t local = kTRU
E);
/// Changes text color.
/// If local is true color is changed locally, otherwise - globa
lly.

   virtual void  SetTextFont(const char *fontName, Bool_t local
= kTRUE);
/// Changes text font specified by name.
/// If local is true font is changed locally - otherwise globall
y.

   virtual void  SetTextFont(FontStruct_t font, Bool_t local = k
TRUE);
/// Changes text font.
/// If local is true font is changed locally - otherwise globall
y.


   GContext_t GetNormGC() const { return fNormGC; }
   FontStruct_t GetFontStruct() const { return fFontStruct; }

   virtual const char *GetTitle() const { return fText->GetStrin
g(); }
   Bool_t HasOwnFont() const;
/// Returns kTRUE if text attributes are unique,
/// returns kFALSE if text attributes are shared (global).

   virtual void  SavePrimitive(std::ostream &out, Option_t *opti
on = "");
/// Save a group frame widget as a C++ statement(s) on output st
ream out.
```

TGHeaderFrame

```
   TGHeaderFrame(const TGWindow *p = 0, UInt_t w = 1, UInt_t h =
1,
                 UInt_t options = kChildFrame,
                 Pixel_t back = GetDefaultFrameBackground());///
 Header Frame constructor.

   virtual Bool_t HandleButton(Event_t* event);/// Handle mouse
button event in header frame.
   virtual Bool_t HandleMotion(Event_t* event);/// Handle mouse
motion events in header frame.
   virtual Bool_t HandleDoubleClick(Event_t *event);/// Handle d
ouble click mouse event in header frame.

   void SetColumnsInfo(Int_t nColumns, TGTextButton  **colHeader
, TGVFileSplitter  **splitHeader);
/// Set columns information in the header frame.
```

code

```
// TGCompositeFrame::SetEditable(Bool_t on)

TGMainFrame *m = new TGMainFrame(gClient->GetRoot(), 500, 500);
m->SetEditable();
gSystem->Load("$ROOTSYS/test/Aclock"); // load Aclock demo
Aclock a;
gROOT->Macro("$ROOTSYS/tutorials/gui/guitest.C");
m->SetEditable(0);
m->MapWindow();
```

```
// TGPicture *TGMainFrame::SetIconPixmap(const char *iconName)

main_frame->SetIconPixmap("/home/root/icons/bld_rgb.png");


// void TGMainFrame::SetIconPixmap(char **xpm_array)

#include "/home/root/icons/bld_rgb.xpm"
//bld_rgb.xpm contains char *bld_rgb[] array
main_frame->SetIconPixmap(bld_rgb);
```

example

TGGC

TGIcon

TGIcon 继承 TGFrame

This class handles GUI icons.

class

```
   TGIcon(const TGWindow *p, const TGPicture *pic, UInt_t w, UIn
t_t h,
      UInt_t options = kChildFrame, Pixel_t back = GetDefaultFra
meBackground()) :
         TGFrame(p, w, h, options, back), fPic(pic), fImage(0),
fPath() { SetWindowName(); }

   TGIcon(const TGWindow *p = 0, const char *image = 0);/// Crea
te icon.

   virtual ~TGIcon();/// Delete icon and free picture.

   virtual void Reset();         //*MENU*
/// Reset icon to original image. It can be used only via contex
t menu.

   const TGPicture *GetPicture() const { return fPic; }
   TImage *GetImage() const { return fImage; }
   virtual void SetPicture(const TGPicture *pic);/// Set icon pi
cture.
   virtual void SetImage(const char *img);/// Set icon image.
   virtual void SetImage(TImage *img);/// Change icon image.
   virtual void SetImagePath(const char *path);
/// Set directory where image is located

   virtual void Resize(UInt_t w = 0, UInt_t h = 0);/// Resize.
   virtual void Resize(TGDimension size) { Resize(size.fWidth, s
ize.fHeight); }
   virtual void MoveResize(Int_t x, Int_t y, UInt_t w = 0, UInt_
```

```
t h = 0);
/// Move icon to (x,y) and resize it to (w,h).

   virtual void ChangeBackgroundColor() { }

   virtual TGDimension GetDefaultSize() const;/// Return size of
 icon.
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save an icon widget as a C++ statement(s) on output stream o
ut.
```

code

example

TGImageMap

TGRegion 继承 TObject
Describes a region

TGRegionWithId 继承 TGRegion
Region with id, tooltip text and popup menu

TGImageMap 继承 TGPictureButton
Clickable image (like MAP in HTML)

class

TGRegion

```
   enum ERegionType { kRectangle, kEllipse };

   TGRegion();
   TGRegion(Int_t x, Int_t y, UInt_t w, UInt_t h, ERegionType =
kRectangle);
   TGRegion(Int_t n, TPoint *points, Bool_t winding = kFALSE);
   TGRegion(Int_t n, Int_t *x, Int_t *y, Bool_t winding = kFALSE
);
   TGRegion(const TArrayS &x, const TArrayS &y, Bool_t winding =
 kFALSE);
   TGRegion(const TGRegion &reg);
   virtual ~TGRegion();

   Bool_t      Contains(const TPoint &p) const;
   Bool_t      Contains(Int_t x, Int_t y) const;
   TGRegion    Unite(const TGRegion &r) const;
   TGRegion    Intersect(const TGRegion &r) const;
   TGRegion    Subtract(const TGRegion &r) const;
   TGRegion    Eor(const TGRegion &r) const;
   TGDimension GetDimension() const;
   TGPosition  GetPosition() const;
   Bool_t      IsNull() const;
```

```
   Bool_t      IsEmpty() const;

   TGRegion operator|(const TGRegion &r) const { return Unite(r)
; }
   TGRegion operator+(const TGRegion &r) const { return Unite(r)
; }
   TGRegion operator&(const TGRegion &r) const { return Intersec
t(r); }
   TGRegion operator-(const TGRegion &r) const { return Subtract
(r); }
   TGRegion operator^(const TGRegion &r) const { return Eor(r);
}
   TGRegion& operator|=(const TGRegion &r) { return *this = *this
 | r; }
   TGRegion& operator+=(const TGRegion &r) { return *this = *this
 + r; }
   TGRegion& operator&=(const TGRegion &r) { return *this = *this
 & r; }
   TGRegion& operator-=(const TGRegion &r) { return *this = *this
 - r; }
   TGRegion& operator^=(const TGRegion &r) { return *this = *this
 ^ r; }
   Bool_t operator==(const TGRegion &r)  const;
   Bool_t operator!=(const TGRegion &r) const { return !(operator
==(r)); }
   TGRegion &operator=(const TGRegion &r);
```

TGRegionWithId

```
   TGRegionWithId();
   TGRegionWithId(Int_t id, Int_t x, Int_t y, UInt_t w, UInt_t h
,
                  ERegionType = kRectangle);
   TGRegionWithId(Int_t id, Int_t n, TPoint *points, Bool_t wind
ing = kFALSE);
   TGRegionWithId(const TGRegionWithId &reg);
   TGRegionWithId(const TGRegion &reg, Int_t id);
   virtual ~TGRegionWithId();

   Int_t        GetId() const { return fId; }
   TGToolTip   *GetToolTipText() const { return fTip; }
   void         SetToolTipText(const char *text, Long_t delayms,
                               const TGFrame *frame);
   TGPopupMenu *GetPopup() const { return fPopup; }
   void         SetPopup(TGPopupMenu *popup) { fPopup = popup; }
   void         DisplayPopup();
```

TGImageMap

```
   enum ENavMode { kNavRegions, kNavGrid };

   TGImageMap(const TGWindow *p = 0, const TGPicture *pic = 0);
   TGImageMap(const TGWindow *p, const TString &pic);
   virtual ~TGImageMap();

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleDoubleClick(Event_t *event);
   virtual Bool_t HandleMotion(Event_t *event);

   ENavMode       GetNavMode() { return fNavMode; }
   void           AddRegion(const TGRegion &region, Int_t id);
   TGPopupMenu   *CreatePopup(Int_t id);
   TGPopupMenu   *GetPopup(Int_t id);

   void SetToolTipText(const char *text, Long_t delayms = 300);
   void SetToolTipText(Int_t id, const char *text, Long_t delaym
s = 300);
   void SetCursor(ECursor cursor = kHand) { fCursorMouseOver = c
ursor; }
   void SetPicture(const TGPicture * /*new_pic*/) { } // disabled


   virtual void RegionClicked(Int_t id); // *SIGNAL*
   virtual void DoubleClicked(Int_t id); // *SIGNAL*
   virtual void DoubleClicked();         // *SIGNAL*
   virtual void OnMouseOver(Int_t id);   // *SIGNAL*
   virtual void OnMouseOut(Int_t id);    // *SIGNAL*
```

code

```
#include "TGImageMap.h"
// TGImageMap
// 插入图片
TGImageMap* fImageMap;
fImageMap = new TGImageMap(frame, "picture.jpg");
frame->AddFrame(fImageMap);
fImageMap->Connect("RegionClicked(Int_t)", "WorldMap", this,"Pri
ntCode(Int_t)");
```

example

TGLabel

TGLabel 继承 TGFrame

class

```
   static FontStruct_t  GetDefaultFontStruct();/// Static return
ing label default font struct.
   static const TGGC   &GetDefaultGC();/// Static returning labe
l default graphics context.

   TGLabel(const TGWindow *p, TGString *text,
           GContext_t norm = GetDefaultGC()(),
           FontStruct_t font = GetDefaultFontStruct(),
           UInt_t options = kChildFrame,
           Pixel_t back = GetDefaultFrameBackground());
/// Create a label GUI object. TGLabel will become the owner of
the
/// text and will delete it in its dtor.

   TGLabel(const TGWindow *p = 0, const char *text = 0,
           GContext_t norm = GetDefaultGC()(),
           FontStruct_t font = GetDefaultFontStruct(),
           UInt_t options = kChildFrame,
           Pixel_t back = GetDefaultFrameBackground());
/// Create a label GUI object.

   virtual ~TGLabel();/// Delete label.

   virtual TGDimension GetDefaultSize() const;/// Return default
 size.

   const TGString *GetText() const { return fText; }
   virtual const char *GetTitle() const { return fText->Data();
}
   virtual void SetText(TGString *newText);
/// Set new text in label. After calling this method one needs t
```

```
o call
/// the parents frame's Layout() method to force updating of the
 label size.
/// The new_text is adopted by the TGLabel and will be properly
deleted.

   void SetText(const char *newText) { SetText(new TGString(newT
ext)); }
   virtual void ChangeText(const char *newText) { SetText(newTex
t); } //*MENU*icon=bld_rename.png*
   virtual void SetTitle(const char *label) { SetText(label); }
   void  SetText(Int_t number) { SetText(new TGString(number));
}
   void  SetTextJustify(Int_t tmode);
/// Set text justification. Mode is an OR of the bits:
/// kTextTop, kTextBottom, kTextLeft, kTextRight, kTextCenterX a
nd
/// kTextCenterY.

   Int_t GetTextJustify() const { return fTMode; }
   virtual void SetTextFont(TGFont *font, Bool_t global = kFALSE)
;
/// Changes text font specified by pointer to TGFont object.
/// If global is true font is changed globally - otherwise local
ly.

   virtual void SetTextFont(FontStruct_t font, Bool_t global = k
FALSE);
/// Changes text font.
/// If global is true font is changed globally - otherwise local
ly.

   virtual void SetTextFont(const char *fontName, Bool_t global
= kFALSE);
/// Changes text font specified by name.
/// If global is true font is changed globally - otherwise local
ly.

   virtual void SetTextColor(Pixel_t color, Bool_t global = kFAL
SE);
```

```
/// Changes text color.
/// If global is true color is changed globally - otherwise loca
lly.

   virtual void SetTextColor(TColor *color, Bool_t global = kFAL
SE);
/// Changes text color.
/// If global is true color is changed globally - otherwise loca
lly.

   virtual void SetForegroundColor(Pixel_t fore) { SetTextColor(
fore); }
   virtual void Disable(Bool_t on = kTRUE)
               { fDisabled = on; fClient->NeedRedraw(this); } //
*TOGGLE* *GETTER=IsDisabled
   virtual void Enable() { fDisabled = kFALSE; fClient->NeedRedr
aw(this); }
   Bool_t IsDisabled() const { return fDisabled; }
   Bool_t HasOwnFont() const;
/// Returns kTRUE if text attributes are unique.
/// Returns kFALSE if text attributes are shared (global).

   void  SetWrapLength(Int_t wl) { fWrapLength = wl; Layout(); }
   Int_t GetWrapLength() const { return fWrapLength; }

   void  Set3DStyle(Int_t style) { f3DStyle = style; fClient->Ne
edRedraw(this); }
   Int_t Get3DStyle() const { return f3DStyle; }

   void SetMargins(Int_t left=0, Int_t right=0, Int_t top=0, Int
_t bottom=0)
      { fMLeft = left; fMRight = right; fMTop = top; fMBottom =
bottom; }
   Int_t GetLeftMargin() const { return fMLeft; }
   Int_t GetRightMargin() const { return fMRight; }
   Int_t GetTopMargin() const { return fMTop; }
   Int_t GetBottomMargin() const { return fMBottom; }

   GContext_t GetNormGC() const { return fNormGC; }
   FontStruct_t GetFontStruct() const { return fFont->GetFontStr
```

```
uct(); }
   TGFont      *GetFont() const  { return fFont; }

   virtual void Layout();
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a label widget as a C++ statement(s) on output stream o
ut.
```

code

```
#include "TGLabel.h"

// TGLabel
const char gReadyMsg[] = "Ready. You can drag list tree items to
 any \
pad in the canvas, or to the \"Base\" folder of the list tree it
self...";
TGLabel *fStatus = new TGLabel(frame, new TGHotString(gReadyMsg)
);
fStatus->SetTextJustify(kTextLeft);
fStatus->SetTextColor(0x0000ff);
fStatus->Enable();
// fStatus->Disable();
// if (fStatus->IsDisabled()) ;
// fStatus->SetText("XXX");
// fStatus->SetText(125);
// fStatus->SetFont("XXX");
fStatus->SetText(Form("abc%ld",100);
frame->AddFrame(fStatus, new TGLayoutHints(kLHintsExpandX | kLHi
ntsCenterY,10, 10, 10, 10));
```

example

TGLayout

A number of different layout classes (TGLayoutManager, TGVerticalLayout, TGHorizontalLayout, TGLayoutHints, etc.).

TGLayoutHints : public TObject, TRefCnt , friend TGFrameElement , TGCompositeFrame This class describes layout hints used by the layout classes.

TGFrameElement : public TObject
Base class used in GUI containers

TGLayoutManager : public TObject
Frame layout manager. This is an abstract class.

TGVerticalLayout : public TGLayoutManager
TGVerticalLayout and TGHorizontalLayout managers.

TGHorizontalLayout : public TGVerticalLayout

TGRowLayout : public TGVerticalLayout
The follwing two layout managers do not make use of TGLayoutHints.

TGColumnLayout : public TGRowLayout

TGMatrixLayout : public TGLayoutManager
This layout managers does not make use of TGLayoutHints.

TGTileLayout : public TGLayoutManager
This are layout managers for the TGListView widget.

TGListLayout : public TGTileLayout

TGListDetailsLayout : public TGTileLayout

class

```
//---- layout hints

enum ELayoutHints {
   kLHintsNoHints = 0,
   kLHintsLeft    = BIT(0),
   kLHintsCenterX = BIT(1),
   kLHintsRight   = BIT(2),
   kLHintsTop     = BIT(3),
   kLHintsCenterY = BIT(4),
   kLHintsBottom  = BIT(5),
   kLHintsExpandX = BIT(6),
   kLHintsExpandY = BIT(7),
   kLHintsNormal  = (kLHintsLeft | kLHintsTop)
   // bits 8-11 used by ETableLayoutHints
};
```

TGLayoutHints

```
   TGLayoutHints(ULong_t hints = kLHintsNormal,
                 Int_t padleft = 0, Int_t padright = 0,
                 Int_t padtop = 0, Int_t padbottom = 0):
     fFE(0), fPrev(0), fLayoutHints(hints), fPadtop(padtop), fPa
dbottom(padbottom),
     fPadleft(padleft), fPadright(padright)
     { SetRefCount(0); }

   TGLayoutHints(const TGLayoutHints &lh);

   virtual ~TGLayoutHints();

   ULong_t GetLayoutHints() const { return fLayoutHints; }
   Int_t   GetPadTop() const { return fPadtop; }
   Int_t   GetPadBottom() const { return fPadbottom; }
   Int_t   GetPadLeft() const { return fPadleft; }
   Int_t   GetPadRight() const { return fPadright; }

   virtual void SetLayoutHints(ULong_t lh) { fLayoutHints = lh;
}
   virtual void SetPadTop(Int_t v)  {  fPadtop = v; }
   virtual void SetPadBottom(Int_t v)  {  fPadbottom = v; }
   virtual void SetPadLeft(Int_t v)  {  fPadleft = v; }
   virtual void SetPadRight(Int_t v)  {  fPadright = v; }

   void Print(Option_t* option = "") const;
   void ls(Option_t* option = "") const { Print(option); }

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

TGFrameElement

```
   TGFrame         *fFrame;     // frame used in layout
   Int_t            fState;     // EFrameState defined in TGFrame.h

   TGLayoutHints   *fLayout;    // layout hints used in layout

   TGFrameElement() : fFrame(0), fState(0), fLayout(0) { }
   TGFrameElement(TGFrame *f, TGLayoutHints *l);
   ~TGFrameElement();

   void Print(Option_t* option = "") const;
   void ls(Option_t* option = "") const { Print(option); }
```

TGLayoutManager

```
   TGLayoutManager() : fModified(kTRUE) {}

   virtual void Layout() = 0;
   virtual TGDimension GetDefaultSize() const = 0;
   virtual void SetDefaultWidth(UInt_t /* w */) {}
   virtual void SetDefaultHeight(UInt_t /* h */) {}
   virtual Bool_t IsModified() const { return fModified; }
   virtual void   SetModified(Bool_t flag = kTRUE) { fModified =
 flag; }
```

TGVerticalLayout

```
   TGVerticalLayout(TGCompositeFrame *main);

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
 ;
```

TGHorizontalLayout

```
   TGHorizontalLayout(TGCompositeFrame *main) : TGVerticalLayout
(main) { }

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGRowLayout

```
   Int_t   fSep;             // interval between frames

   TGRowLayout(TGCompositeFrame *main, Int_t s = 0) :
      TGVerticalLayout(main), fSep(s) { }

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGColumnLayout

```
   TGColumnLayout(TGCompositeFrame *main, Int_t s = 0) : TGRowLa
yout(main, s) { }

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGMatrixLayout

```
   Int_t   fSep;                      // interval between frames
   Int_t   fHints;                    // layout hints (currently
 not used)
   UInt_t  fRows;                     // number of rows
   UInt_t  fColumns;                  // number of columns

   TGMatrixLayout(TGCompositeFrame *main, UInt_t r, UInt_t c, In
t_t s=0, Int_t h=0);

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGTileLayout

```
   TGTileLayout(TGCompositeFrame *main, Int_t sep = 0);

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual Bool_t IsModified() const { return fModified; }
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGListLayout

```
   TGListLayout(TGCompositeFrame *main, Int_t sep = 0) :
      TGTileLayout(main, sep) { }

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

TGListDetailsLayout

```
   TGListDetailsLayout(TGCompositeFrame *main, Int_t sep = 0, UI
nt_t w = 0) :
      TGTileLayout(main, sep), fWidth(w) { }

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SetDefaultWidth(UInt_t w) { fWidth = w; }
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
```

code

example

TGListBox

TGListBox 继承 TGCompositeFrame, TGWidget

A TGListBox widget.

A listbox is a box, possibly with scrollbar, containing entries.
Currently entries are simple text strings (TGTextLBEntry).
A TGListBox looks a lot like a TGCanvas. It has a TGViewPort containing a TGLBContainer which contains the entries and it also has a vertical scrollbar which becomes visible if there are more items than fit in the visible part of the container.

```
Selecting an item in the listbox will generate the event:
kC_COMMAND, kCM_LISTBOX, listbox id, item id.
```

class

```
   TGListBox(const TGWindow *p = 0, Int_t id = -1,
             UInt_t options = kSunkenFrame | kDoubleBorder,
             Pixel_t back = GetWhitePixel());
   virtual ~TGListBox();

   virtual void AddEntry(TGString *s, Int_t id);
   virtual void AddEntry(const char *s, Int_t id);
   virtual void AddEntry(TGLBEntry *lbe, TGLayoutHints *lhints);
   virtual void AddEntrySort(TGString *s, Int_t id);
   virtual void AddEntrySort(const char *s, Int_t id);
   virtual void AddEntrySort(TGLBEntry *lbe, TGLayoutHints *lhin
ts);
   virtual void InsertEntry(TGString *s, Int_t id, Int_t afterID)
;
   virtual void InsertEntry(const char *s , Int_t id, Int_t afte
rID);
   virtual void InsertEntry(TGLBEntry *lbe, TGLayoutHints *lhint
s, Int_t afterID);
```

```
   virtual void NewEntry(const char *s = "Entry");             /
/*MENU*
   virtual void RemoveEntry(Int_t id = -1);                    /
/*MENU*
   virtual void RemoveAll();                                   /
/*MENU*
   virtual void RemoveEntries(Int_t from_ID, Int_t to_ID);
   virtual void ChangeBackground(Pixel_t back);
   virtual void SetTopEntry(Int_t id = -1);
   virtual void SetMultipleSelections(Bool_t multi = kTRUE)
                  { fLbc->SetMultipleSelections(multi); }      /
/*TOGGLE* *GETTER=GetMultipleSelections
   virtual Bool_t GetMultipleSelections() const
                  { return fLbc->GetMultipleSelections(); }
   virtual Int_t  GetNumberOfEntries() const
                  { return fLbc->GetList()->GetSize(); }
   virtual TGLBEntry    *GetEntry(Int_t id) const;
   virtual TGLBEntry    *FindEntry(const char *s) const;
   virtual TGFrame      *GetContainer() const { return fVport->G
etContainer(); }
   virtual TGViewPort   *GetViewPort() const { return fVport; }
   virtual TGScrollBar  *GetScrollBar() const { return fVScrollb
ar; }
   virtual TGVScrollBar *GetVScrollbar() const { return fVScroll
bar; }

   virtual void DrawBorder();
   virtual void Resize(UInt_t w, UInt_t h);
   virtual void Resize(TGDimension size) { Resize(size.fWidth, s
ize.fHeight); }
   virtual void MoveResize(Int_t x, Int_t y, UInt_t w, UInt_t h)
;
   virtual void Layout();
   virtual void SetLayoutManager(TGLayoutManager*) { }
   virtual void SortByName(Bool_t ascend = kTRUE);   //*MENU*ico
n=bld_sortup.png*
   virtual void IntegralHeight(Bool_t mode) { fIntegralHeight =
mode; }
   virtual TGDimension GetDefaultSize() const;
```

```
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);

   virtual TGLBEntry *Select(Int_t id, Bool_t sel = kTRUE)
                                          { return fLbc->Select(id,
 sel); }
   virtual Int_t GetSelected() const;
   virtual Bool_t GetSelection(Int_t id) { return fLbc->GetSelec
tion(id); }
   virtual TGLBEntry *GetSelectedEntry() const { return fLbc->Ge
tSelectedEntry(); }
   virtual void GetSelectedEntries(TList *selected);
   UInt_t  GetItemVsize() const { return fItemVsize; }

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");

   virtual void Selected(Int_t widgetId, Int_t id);   //*SIGNAL*
   virtual void Selected(Int_t id) { Emit("Selected(Int_t)", id)
; } //*SIGNAL*
   virtual void Selected(const char *txt) { Emit("Selected(char*
)", txt); } //*SIGNAL
   virtual void DoubleClicked(Int_t widgetId, Int_t id);   //*SI
GNAL*
   virtual void DoubleClicked(Int_t id) { Emit("DoubleClicked(In
t_t)", id); } //*SIGNAL*
   virtual void DoubleClicked(const char *txt) { Emit("DoubleCli
cked(char*)", txt); } //*SIGNAL
   virtual void SelectionChanged() { Emit("SelectionChanged()");
 } //*SIGNAL*
```

code

```
#include "TGListBox.h"
```

```
// TGListBox
// 可选择列表，可单选、多选
TGListBox           *fListBox;
TList               *fSelected;
fListBox = new TGListBox(frame, 89);
fSelected = new TList;
char tmp[20];
for (int i = 0; i < 20; ++i) {
  sprintf(tmp, "Entry %i", i+1);
  fListBox->AddEntry(tmp, i+1/*IDs*/);
}
fListBox->Resize(100,150);
frame->AddFrame(fListBox, new TGLayoutHints(kLHintsTop | kLHints
Left | kLHintsExpandX | kLHintsExpandY, 5, 5, 5, 5));
fListBox->SetMultipleSelections(0/*0  1*/);//设置是否可多选
fSelected->Clear();// Writes selected entries in TList if multis
election.
if (fListBox->GetMultipleSelections()) {
  Printf("Selected entries are:\n");
  fListBox->GetSelectedEntries(fSelected);
  fSelected->ls();
 } else {
  Printf("Selected entries is: %d\n", fListBox->GetSelected());
 }
if (fSelected) {//不用之后
  fSelected->Delete();
  delete fSelected;
 }
```

example

TGListTree

A list tree is a widget that can contain a number of items arranged in a tree structure. The items are represented by small folder icons that can be either open or closed.

The TGListTree is user callable. The TGListTreeItem is a service class of the list tree.

A list tree can generate the following events:
kC_LISTTREE, kCT_ITEMCLICK, which button, location (y<<16|x).
kC_LISTTREE, kCT_ITEMDBLCLICK, which button, location (y<<16|x).

TGListTreeItem friend TGListTree
Abstract base-class for items that go into a TGListTree container.

TGListTreeItemStd 继承 TGListTreeItem
Item that goes into a TGListTree container

TGListTree 继承 TGContainer
Show items in a tree structured list

class

TGListTreeItem

```
    TGListTreeItem(TGClient *client=gClient);
    virtual ~TGListTreeItem() {}

    TGListTreeItem *GetParent()      const { return fParent; }
    TGListTreeItem *GetFirstChild()  const { return fFirstchild;
}
    TGListTreeItem *GetLastChild()   const { return fLastchild;
}
    TGListTreeItem *GetPrevSibling() const { return fPrevsibling;
 }
    TGListTreeItem *GetNextSibling() const { return fNextsibling;
```

```
 }

   virtual Bool_t          IsOpen()    const { return fOpen; }
   virtual void            SetOpen(Bool_t o) { fOpen = o; }

   virtual Bool_t          IsActive() const = 0;
   virtual Pixel_t         GetActiveColor() const = 0;
   virtual void            SetActive(Bool_t) {}

   void                    Rename(const char* new_name) { SetTex
t(new_name); }
   virtual const char     *GetText() const = 0;
   virtual Int_t           GetTextLength() const = 0;
   virtual const char     *GetTipText() const = 0;
   virtual Int_t           GetTipTextLength() const = 0;
   virtual void            SetText(const char *) {}
   virtual void            SetTipText(const char *) {}

   virtual void            SetUserData(void *, Bool_t=kFALSE) {}
   virtual void           *GetUserData() const = 0;

   virtual const TGPicture*GetPicture() const = 0;
   virtual void            SetPictures(const TGPicture*, const T
GPicture*) {}
   virtual const TGPicture*GetCheckBoxPicture() const = 0;
   virtual void            SetCheckBoxPictures(const TGPicture*,
const TGPicture*) {}
   virtual UInt_t          GetPicWidth() const;

   virtual void            SetCheckBox(Bool_t=kTRUE) {}
   virtual Bool_t          HasCheckBox() const = 0;
   virtual void            CheckItem(Bool_t=kTRUE) = 0;
   virtual void            Toggle() { SetCheckBox( ! IsChecked()
); }
   virtual Bool_t          IsChecked() const = 0;

   // Propagation of checked-state form children to parents.
   virtual void            CheckAllChildren(Bool_t=kTRUE) {}
   virtual void            CheckChildren(TGListTreeItem*, Bool_t)
 {}
```

```
   virtual Bool_t          HasCheckedChild(Bool_t=kFALSE)   { re
turn kTRUE; } // !!!!
   virtual Bool_t          HasUnCheckedChild(Bool_t=kFALSE) { re
turn kTRUE; } // !!!!
   virtual void            UpdateState() {}

   // Item coloration (underline + minibox)
   virtual Bool_t          HasColor() const = 0;
   virtual Color_t         GetColor() const = 0;
   virtual void            SetColor(Color_t) {}
   virtual void            ClearColor() {}

   // Drag and drop.
   void            SetDNDSource(Bool_t onoff)
                   { if (onoff) fDNDState |= kIsDNDSource; else
fDNDState &= ~kIsDNDSource; }
   void            SetDNDTarget(Bool_t onoff)
                   { if (onoff) fDNDState |= kIsDNDTarget; else
fDNDState &= ~kIsDNDTarget; }
   Bool_t          IsDNDSource() const { return fDNDState & kIsD
NDSource; }
   Bool_t          IsDNDTarget() const { return fDNDState & kIsD
NDTarget; }

   // Allow handling by the items themselves ... NOT USED in TGL
istTree yet !!!!
   virtual Bool_t  HandlesDragAndDrop() const { return kFALSE; }
   virtual void    HandleDrag() {}
   virtual void    HandleDrop() {}

   virtual void    SavePrimitive(std::ostream&, Option_t*, Int_t)
 {}
```

TGListTreeItemStd

```
   TGListTreeItemStd(TGClient *fClient = gClient, const char *na
me = 0,
                     const TGPicture *opened = 0, const TGPictur
e *closed = 0,
```

```
                        Bool_t checkbox = kFALSE);
   virtual ~TGListTreeItemStd();

   virtual Pixel_t         GetActiveColor() const;
   virtual Bool_t          IsActive()       const { return fActi
ve; }
   virtual void            SetActive(Bool_t a)    { fActive = a;
 }

   virtual const char     *GetText()             const { return
fText.Data(); }
   virtual Int_t           GetTextLength()       const { return
fText.Length(); }
   virtual const char     *GetTipText()          const { return
fTipText.Data(); }
   virtual Int_t           GetTipTextLength()    const { return
fTipText.Length(); }
   virtual void            SetText(const char *text)   { fText =
 text; }
   virtual void            SetTipText(const char *tip) { fTipTex
t = tip; }

   virtual void            SetUserData(void *userData, Bool_t ow
n=kFALSE) { fUserData = userData; fOwnsData=own; }
   virtual void           *GetUserData() const { return fUserDat
a; }

   virtual const TGPicture*GetPicture()         const { return f
Open ? fOpenPic : fClosedPic; }
   virtual const TGPicture*GetCheckBoxPicture() const { return f
CheckBox ? (fChecked ? fCheckedPic : fUncheckedPic) : 0; }
   virtual void            SetPictures(const TGPicture *opened,
const TGPicture *closed);
   virtual void            SetCheckBoxPictures(const TGPicture *
checked, const TGPicture *unchecked);

   virtual void            SetCheckBox(Bool_t on = kTRUE);
   virtual Bool_t          HasCheckBox() const { return fCheckBo
x; }
   virtual void            CheckItem(Bool_t checked = kTRUE) { f
```

```
Checked = checked; }
   virtual void            Toggle() { fChecked = !fChecked; }
   virtual Bool_t          IsChecked() const { return fChecked;
}

   virtual void            CheckAllChildren(Bool_t state = kTRUE)
;
   virtual void            CheckChildren(TGListTreeItem *item, B
ool_t state);
   virtual Bool_t          HasCheckedChild(Bool_t first=kFALSE);
   virtual Bool_t          HasUnCheckedChild(Bool_t first=kFALSE)
;
   virtual void            UpdateState();

   virtual Bool_t          HasColor() const { return fHasColor;
}
   virtual Color_t         GetColor() const { return fColor; }
   virtual void            SetColor(Color_t color) { fHasColor =
true;fColor = color; }
   virtual void            ClearColor() { fHasColor = false; }

   virtual void            SavePrimitive(std::ostream &out, Opti
on_t *option, Int_t n);
```

```
   //---- color markup mode of tree items
   enum EColorMarkupMode {
      kDefault        = 0,
      kColorUnderline = BIT(0),
      kColorBox       = BIT(1)
   };
   enum ECheckMode {
      kSimple    = BIT(2),
      kRecursive = BIT(3)
   };

   TGListTree(TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
              UInt_t options = 0, Pixel_t back = GetWhitePixel()
```

```
);
   TGListTree(TGCanvas *p, UInt_t options, Pixel_t back = GetWhi
tePixel());

   virtual ~TGListTree();

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleDoubleClick(Event_t *event);
   virtual Bool_t HandleCrossing(Event_t *event);
   virtual Bool_t HandleMotion(Event_t *event);
   virtual Bool_t HandleKey(Event_t *event);

   virtual void SetCanvas(TGCanvas *canvas) { fCanvas = canvas;
}
   virtual void DrawRegion(Int_t x, Int_t y, UInt_t w, UInt_t h)
;

   virtual void DrawOutline(Handle_t id, TGListTreeItem *item, P
ixel_t col=0xbbbbbb,
                            Bool_t clear=kFALSE);
   virtual void DrawActive(Handle_t id, TGListTreeItem *item);

   virtual TGDimension GetDefaultSize() const
            { return TGDimension(fDefw, fDefh); }

   void            AddItem(TGListTreeItem *parent, TGListTreeIte
m *item);
   TGListTreeItem *AddItem(TGListTreeItem *parent, const char *s
tring,
                           const TGPicture *open = 0,
                           const TGPicture *closed = 0,
                           Bool_t checkbox = kFALSE);
   TGListTreeItem *AddItem(TGListTreeItem *parent, const char *s
tring,
                           void *userData, const TGPicture *open
 = 0,
                           const TGPicture *closed = 0,
                           Bool_t checkbox = kFALSE);
   void  RenameItem(TGListTreeItem *item, const char *string);
   Int_t DeleteItem(TGListTreeItem *item);
```

```
   void  OpenItem(TGListTreeItem *item);
   void  CloseItem(TGListTreeItem *item);
   void  CheckItem(TGListTreeItem *item, Bool_t check = kTRUE);
   void  SetCheckBox(TGListTreeItem *item, Bool_t on = kTRUE);
   void  ToggleItem(TGListTreeItem *item);
   Int_t RecursiveDeleteItem(TGListTreeItem *item, void *userDat
a);

   Int_t DeleteChildren(TGListTreeItem *item);
   Int_t Reparent(TGListTreeItem *item, TGListTreeItem *newparen
t);
   Int_t ReparentChildren(TGListTreeItem *item, TGListTreeItem *
newparent);
   void  SetToolTipItem(TGListTreeItem *item, const char *string)
;
   void  SetAutoTips(Bool_t on = kTRUE) { fAutoTips = on; }
   void  SetAutoCheckBoxPic(Bool_t on) { fAutoCheckBoxPic = on;
}
   void  SetSelected(TGListTreeItem *item) { fSelected = item; }
   void  AdjustPosition(TGListTreeItem *item);
   void  AdjustPosition() { TGContainer::AdjustPosition(); }

   // overwrite TGContainer's methods
   void Home(Bool_t select = kFALSE);
   void End(Bool_t select = kFALSE);
   void PageUp(Bool_t select = kFALSE);
   void PageDown(Bool_t select = kFALSE);
   void LineUp(Bool_t select = kFALSE);
   void LineDown(Bool_t select = kFALSE);
   void Search(Bool_t close = kTRUE);

   Int_t Sort(TGListTreeItem *item);
   Int_t SortSiblings(TGListTreeItem *item);
   Int_t SortChildren(TGListTreeItem *item);
   void  HighlightItem(TGListTreeItem *item);
   void  ClearHighlighted();
   void  GetPathnameFromItem(TGListTreeItem *item, char *path, I
nt_t depth = 0);
   void  UnselectAll(Bool_t draw);
   void  SetToolTipText(const char *text, Int_t x, Int_t y, Long
```

```
_t delayms);
   void  HighlightItem(TGListTreeItem *item, Bool_t state, Bool_
t draw);
   void  HighlightChildren(TGListTreeItem *item, Bool_t state, B
ool_t draw);
   void  DisableOpen(Bool_t disable = kTRUE) { fDisableOpen = di
sable;}
   void  GetChecked(TList *checked);
   void  GetCheckedChildren(TList *checked, TGListTreeItem *item)
;
   void  CheckAllChildren(TGListTreeItem *item, Bool_t state);

   TGListTreeItem *GetFirstItem()  const { return fFirst; }
   TGListTreeItem *GetSelected()   const { return fSelected; }
   TGListTreeItem *GetCurrent()    const { return fCurrent; }
   TGListTreeItem *GetBelowMouse() const { return fBelowMouse; }
   TGListTreeItem *FindSiblingByName(TGListTreeItem *item, const
char *name);
   TGListTreeItem *FindSiblingByData(TGListTreeItem *item, void
*userData);
   TGListTreeItem *FindChildByName(TGListTreeItem *item, const c
har *name);
   TGListTreeItem *FindChildByData(TGListTreeItem *item, void *u
serData);
   TGListTreeItem *FindItemByPathname(const char *path);
   TGListTreeItem *FindItemByObj(TGListTreeItem *item, void *ptr)
;

   void  AddItem(const char *string) { AddItem(fSelected, string
); } //*MENU*
   void  AddRoot(const char *string) { AddItem(0, string); } //*
MENU*
   Int_t DeleteSelected() { return (fSelected ? DeleteItem(fSele
cted) : 0); } //*MENU*
   void  RenameSelected(const char *string) { if (fSelected) Ren
ameItem(fSelected, string); } //*MENU*

   virtual void MouseOver(TGListTreeItem *entry);  //*SIGNAL*
   virtual void MouseOver(TGListTreeItem *entry, UInt_t mask);
//*SIGNAL*
```

```
   virtual void KeyPressed(TGListTreeItem *entry, UInt_t keysym,
 UInt_t mask);  //*SIGNAL*
   virtual void ReturnPressed(TGListTreeItem *entry);  //*SIGNAL*

   virtual void Clicked(TGListTreeItem *entry, Int_t btn);  //*S
IGNAL*
   virtual void Clicked(TGListTreeItem *entry, Int_t btn, Int_t
x, Int_t y);  //*SIGNAL*
   virtual void Clicked(TGListTreeItem *entry, Int_t btn, UInt_t
 mask, Int_t x, Int_t y);  //*SIGNAL*
   virtual void DoubleClicked(TGListTreeItem *entry, Int_t btn);
  //*SIGNAL*
   virtual void DoubleClicked(TGListTreeItem *entry, Int_t btn,
Int_t x, Int_t y);  //*SIGNAL*
   virtual void Checked(TObject *obj, Bool_t check);  //*SIGNAL*
   virtual void DataDropped(TGListTreeItem *item, TDNDData *data)
;  //*SIGNAL*

   // Utility functions
   Int_t        FontHeight();
   Int_t        FontAscent();
   Int_t        TextWidth(const char *c);

   static const TGPicture *GetOpenPic();
   static const TGPicture *GetClosedPic();
   static const TGPicture *GetCheckedPic();
   static const TGPicture *GetUncheckedPic();

   // User control
   void         SetUserControl(Bool_t ctrl=kTRUE) { fUserControl
led = ctrl; }
   Bool_t       HasUserControl() const { return fUserControlled;
 }
   void         SetEventHandled(Bool_t eh=kTRUE) { fEventHandled
 = eh; }
   Bool_t       IsEventHandled() const { return fEventHandled; }

   Bool_t   HandleDNDDrop(TDNDData *data);
   Atom_t   HandleDNDPosition(Int_t x, Int_t y, Atom_t action,
                              Int_t xroot, Int_t yroot);
```

```
   Atom_t   HandleDNDEnter(Atom_t * typelist);
   Bool_t   HandleDNDLeave();

   virtual TDNDData *GetDNDData(Atom_t) {
      return &fDNDData;
   }

   EColorMarkupMode GetColorMode() const { return fColorMode; }
   void SetColorMode(EColorMarkupMode colorMode) { fColorMode =
colorMode; }

   ECheckMode GetCheckMode() const { return fCheckMode; }
   void SetCheckMode(ECheckMode mode) { fCheckMode = mode; }

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

code

example

TGListView

A list view is a widget that can contain a number of items arranged in a grid or list. The items can be represented either by a string or by an icon.

The TGListView is user callable. The other classes are service classes of the list view.

```
A list view can generate the following events:

kC_CONTAINER, kCT_SELCHANGED, total items, selected items.

kC_CONTAINER, kCT_ITEMCLICK, which button, location (y<<16|x).

kC_CONTAINER, kCT_ITEMDBLCLICK, which button, location (y<<16|x)
.
```

TGLVEntry 继承 TGFrame
Item that goes into a TGListView container

TGListView 继承 TGCanvas
List view widget (iconbox, small icons or tabular view)

TGLVContainer 继承 TGContainer

class

```
enum EListViewMode {
   kLVLargeIcons,
   kLVSmallIcons,
   kLVList,
   kLVDetails
};
```

TGLVEntry

```
   TGLVEntry(const TGWindow *p = 0,
             const TGPicture *bigpic = 0, const TGPicture *small
pic = 0,
             TGString *name = 0, TGString **subnames = 0,
             EListViewMode ViewMode = kLVDetails,
             UInt_t options = kChildFrame,
             Pixel_t back = GetWhitePixel());

   TGLVEntry(const TGLVContainer *p,
             const TString& name, const TString& cname, TGString
 **subnames = 0,
             UInt_t options = kChildFrame, Pixel_t back = GetWhi
tePixel());

   virtual ~TGLVEntry();

   virtual void SetViewMode(EListViewMode viewMode);

   virtual void        Activate(Bool_t a);
   Bool_t              IsActive() const { return fActive; }
   TGString           *GetItemName() const { return fItemName; }
   virtual const char *GetTitle() const { return fItemName->GetS
tring(); }
   virtual void        SetTitle(const char *text) { *fItemName =
 text; }
   void                SetItemName(const char *name) { *fItemNam
e = name; }
   const TGPicture    *GetPicture() const { return fCurrent; }
   EListViewMode       GetViewMode() const { return fViewMode; }
   void                SetUserData(void *userData) { fUserData =
 userData; }
   void               *GetUserData() const { return fUserData; }
   virtual TGString  **GetSubnames() const { return fSubnames; }
   virtual TGString   *GetSubname(Int_t idx) const { if (fSubnam
es) return fSubnames[idx]; else return 0; }
   virtual void        SetSubnames(const char* n1="",const char*
 n2="",const char* n3="",
                                   const char* n4="",const char*
 n5="",const char* n6="",
```

```
                                        const char* n7="",const char*
 n8="",const char* n9="",
                                        const char* n10="",const char
* n11="",const char* n12="");
   virtual void        SetPictures(const TGPicture *bigpic = 0,
const TGPicture *smallpic = 0);
   virtual void        SetColumns(Int_t *cpos, Int_t *jmode) { f
Cpos = cpos; fJmode = jmode; }
   virtual void        SetCheckedEntry(Bool_t check = kTRUE) { f
Checked = check; }

   virtual TGDimension GetDefaultSize() const;
   virtual Int_t       GetSubnameWidth(Int_t idx) const { return
 fCtw[idx]; }

   virtual void DrawCopy(Handle_t id, Int_t x, Int_t y);
```

TGListView

```
   TGListView(const TGWindow *p, UInt_t w, UInt_t h,
              UInt_t options = kSunkenFrame | kDoubleBorder,
              Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGListView();

   virtual void   ResizeColumns();
   virtual void   Layout();
   virtual void   LayoutHeader(TGFrame *head);
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);
   virtual void   ScrollHeader(Int_t pos);
   virtual void   SetContainer(TGFrame *f);
   virtual void   AdjustHeaders() { fJustChanged = kTRUE; Layout
Header(0); }
   virtual void   SetHeaders(Int_t ncolumns);
   virtual void   SetHeader(const char *s, Int_t hmode, Int_t cm
ode, Int_t idx);
   virtual void   SetDefaultHeaders();
   virtual void   SetViewMode(EListViewMode viewMode);
   TGTextButton** GetHeaderButtons() { return fColHeader; }
```

```
   UInt_t          GetNumColumns() { return fNColumns; }
   EListViewMode   GetViewMode() const { return fViewMode; }
   virtual const char *GetHeader(Int_t idx) const;
   virtual void    SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
   virtual void    SetIncrements(Int_t hInc, Int_t vInc);
   virtual void    SetDefaultColumnWidth(TGVFileSplitter* splitte
r);
   TGDimension     GetMaxItemSize() const { return fMaxSize; }

   virtual void SelectionChanged() { Emit("SelectionChanged()");
 }  //*SIGNAL*
   virtual void Clicked(TGLVEntry *entry, Int_t btn);  //*SIGNAL*

   virtual void Clicked(TGLVEntry *entry, Int_t btn, Int_t x, In
t_t y);  //*SIGNAL*
   virtual void DoubleClicked(TGLVEntry *entry, Int_t btn);  //*
SIGNAL*
   virtual void DoubleClicked(TGLVEntry *entry, Int_t btn, Int_t
 x, Int_t y);  //*SIGNAL*
```

TGLVContainer

```
   TGLVContainer(const TGWindow *p, UInt_t w, UInt_t h,
                 UInt_t options = kSunkenFrame,
                 Pixel_t back = GetDefaultFrameBackground());
   TGLVContainer(TGCanvas *p, UInt_t options = kSunkenFrame,
                 Pixel_t back = GetDefaultFrameBackground());

   virtual ~TGLVContainer();

   TGListView  *GetListView() const { return fListView; }

   virtual void AddItem(TGLVEntry *item)
                  { AddFrame(item, fItemLayout); item->SetColumn
s(fCpos, fJmode); fTotal++; }
   virtual void SelectEntry(TGLVEntry *item)
                  { ActivateItem(item->GetFrameElement()); }
```

```
   virtual void  SetListView(TGListView *lv) { fListView = lv; }
   virtual void  RemoveItemWithData(void *userData);
   virtual void  SetViewMode(EListViewMode viewMode);
   EListViewMode GetViewMode() const { return fViewMode; }
   virtual void  SetColumns(Int_t *cpos, Int_t *jmode);

   virtual TGDimension GetPageDimension() const;
   virtual TGDimension GetMaxItemSize() const;
   virtual Int_t GetMaxSubnameWidth(Int_t idx) const;
   virtual void  SetColHeaders(const char* n1="",const char* n2=
"",const char* n3="",
                               const char* n4="",const char* n5=
"",const char* n6="",
                               const char* n7="",const char* n8=
"",const char* n9="",
                               const char* n10="",const char* n1
1="",const char* n12="");
   virtual void LineUp(Bool_t select = kFALSE);
   virtual void LineDown(Bool_t select = kFALSE);
   virtual void LineLeft(Bool_t select = kFALSE);
   virtual void LineRight(Bool_t select = kFALSE);

   virtual Bool_t HandleButton(Event_t* event);
   TList *GetSelectedItems();
   TList *GetSelectedEntries();
   Bool_t GetMultipleSelection() const { return fMultiSelect; };
   void   SetMultipleSelection(Bool_t multi = kTRUE) { fMultiSel
ect = multi; };
   void   SetHeaders(Int_t ncolumns) { fListView->SetHeaders(nco
lumns); }
   void   SetHeader(const char *s, Int_t hmode, Int_t cmode, Int
_t idx)
                              { fListView->SetHeader(s,hmode,cmo
de,idx); }
   void   SetDefaultHeaders() { fListView->SetDefaultHeaders();
}
   const char *GetHeader(Int_t idx) const { return fListView->Ge
tHeader(idx); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
```

code

example

TGMdiDecorFrame

TGMdiFrame

TGMdiMainFrame

TGMdiMenu

TGMenu

TGMenuEntry 继承 TObject , friend TGPopupMenu , TGMenuBar
This class contains all information about a menu entry.
It is a fully protected class used internally by TGPopupMenu.

TGPopupMenu 继承 TGFrame ,friend TGMenuTitle , TGMenuBar , TGSplitButton
This class creates a popup menu object. Popup menu's are attached to TGMenuBar objects.

TGMenuTitle 继承 TGFrame
This class creates a menu title. A menu title is a frame to which a popup menu can be attached. Menu titles are automatically created when adding a popup menu to a menubar.

TGMenuBar 继承 TGHorizontalFrame , friend TGPopupMenu This class creates a menu bar.

class

TGMenuEntry

```
   TGMenuEntry(): fEntryId(0), fUserData(0), fType(), fStatus(0)
,
      fEx(0), fEy(0), fEw(0), fEh(0), fLabel(0), fShortcut(0), f
Pic(0), fPopup(0) { }
   virtual ~TGMenuEntry() { if (fLabel) delete fLabel; if (fShor
tcut) delete fShortcut; }

   Int_t           GetEntryId() const { return fEntryId; }
   const char     *GetName() const { return fLabel ? fLabel->GetS
tring() : 0; }
   const char     *GetShortcutText() const { return fShortcut ? f
Shortcut->GetString() : 0; }
   virtual Int_t  GetStatus() const { return fStatus; }
   EMenuEntryType GetType() const { return fType; }
   TGPopupMenu    *GetPopup() const { return fPopup; }
   TGHotString    *GetLabel() const  { return fLabel; }
   TGString       *GetShortcut() const { return fShortcut; }
   Int_t           GetEx() const { return fEx; }
   Int_t           GetEy() const { return fEy; }
   UInt_t          GetEw() const { return fEw; }
   UInt_t          GetEh() const { return fEh; }
   const TGPicture *GetPic() const { return fPic; }
   void           *GetUserData() const { return fUserData; }
```

TGPopupMenu

```
   TGPopupMenu(const TGWindow *p = 0, UInt_t w = 10, UInt_t h =
10,
               UInt_t options = 0);
   virtual ~TGPopupMenu();

   virtual void AddEntry(TGHotString *s, Int_t id, void *ud = 0,
                         const TGPicture *p = 0, TGMenuEntry *be
fore = 0);
   virtual void AddEntry(const char *s, Int_t id, void *ud = 0,
                         const TGPicture *p = 0, TGMenuEntry *be
fore = 0);
   virtual void AddSeparator(TGMenuEntry *before = 0);
   virtual void AddLabel(TGHotString *s, const TGPicture *p = 0,
```

```
                          TGMenuEntry *before = 0);
   virtual void AddLabel(const char *s, const TGPicture *p = 0,
                          TGMenuEntry *before = 0);
   virtual void AddPopup(TGHotString *s, TGPopupMenu *popup,
                          TGMenuEntry *before = 0, const TGPictur
e *p = 0);
   virtual void AddPopup(const char *s, TGPopupMenu *popup,
                          TGMenuEntry *before = 0, const TGPictur
e *p = 0);
   virtual void   EnableEntry(Int_t id);
   virtual void   DisableEntry(Int_t id);
   virtual Bool_t IsEntryEnabled(Int_t id);
   virtual void   HideEntry(Int_t id);
   virtual Bool_t IsEntryHidden(Int_t id);
   virtual void   DefaultEntry(Int_t id);
   virtual void   CheckEntry(Int_t id);
   virtual void   CheckEntryByData(void *user_data);
   virtual void   UnCheckEntry(Int_t id);
   virtual void   UnCheckEntryByData(void *user_data);
   virtual void   UnCheckEntries();
   virtual Bool_t IsEntryChecked(Int_t id);
   virtual void   RCheckEntry(Int_t id, Int_t IDfirst, Int_t IDl
ast);
   virtual Bool_t IsEntryRChecked(Int_t id);
   virtual void   PlaceMenu(Int_t x, Int_t y, Bool_t stick_mode,
                            Bool_t grab_pointer);
   virtual Int_t  EndMenu(void *&userData);
   virtual void   DeleteEntry(Int_t id);
   virtual void   DeleteEntry(TGMenuEntry *entry);
   virtual TGMenuEntry *GetEntry(Int_t id);
   virtual TGMenuEntry *GetCurrent() const { return fCurrent; }
   virtual TGMenuEntry *GetEntry(const char *s);
   const TList    *GetListOfEntries() const { return fEntryList;
 }
   virtual void    DrawBorder();
   virtual Bool_t  HandleButton(Event_t *event);
   virtual Bool_t  HandleMotion(Event_t *event);
   virtual Bool_t  HandleCrossing(Event_t *event);
   virtual Bool_t  HandleTimer(TTimer *t);
   virtual void    Associate(const TGWindow *w) { fMsgWindow = w
```

```
; }
   virtual void    SetMenuBar(TGMenuBar *bar) { fMenuBar = bar;
}
   TGMenuBar      *GetMenuBar() const { return fMenuBar; }
   virtual void    Activate(Bool_t) { }
   virtual void    Activate(TGMenuEntry *entry);
   virtual void    SavePrimitive(std::ostream &out, Option_t *op
tion = "");

   UInt_t GetEntrySep()  const { return fEntrySep; }
   virtual void SetEntrySep(UInt_t sep)  { fEntrySep = sep; }

   virtual void PoppedUp() { Emit("PoppedUp()"); }
          // *SIGNAL*
   virtual void PoppedDown() { Emit("PoppedDown()"); }
          // *SIGNAL*
   virtual void Highlighted(Int_t id) { Emit("Highlighted(Int_t)"
, id); } // *SIGNAL*
   virtual void Activated(Int_t id) { Emit("Activated(Int_t)", i
d); }     // *SIGNAL*
```

TGMenuTitle

```
   static FontStruct_t  GetDefaultFontStruct();
   static const TGGC   &GetDefaultSelectedGC();
   static const TGGC   &GetDefaultGC();

   TGMenuTitle(const TGWindow *p = 0, TGHotString *s = 0, TGPopu
pMenu *menu = 0,
               GContext_t norm = GetDefaultGC()(),
               FontStruct_t font = GetDefaultFontStruct(),
               UInt_t options = 0);
   virtual ~TGMenuTitle() { if (fLabel) delete fLabel; }

   Pixel_t      GetTextColor() const { return fTextColor; }
   void         SetTextColor(Pixel_t col) { fTextColor = col; }
   virtual void SetState(Bool_t state);
   Bool_t       GetState() const { return fState; }
   Int_t        GetHotKeyCode() const { return fHkeycode; }
   TGPopupMenu *GetMenu() const { return fMenu; }
   const char  *GetName() const { return fLabel ? fLabel->GetStr
ing() : 0; }
   virtual void DoSendMessage();
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

TGMenuBar

```
   TGMenuBar(const TGWindow *p = 0, UInt_t w = 60, UInt_t h = 20
,
             UInt_t options = kHorizontalFrame | kRaisedFrame);
   virtual ~TGMenuBar();

   virtual void AddPopup(TGHotString *s, TGPopupMenu *menu, TGLa
youtHints *l,
                         TGPopupMenu *before = 0);
   virtual void AddPopup(const char *s, TGPopupMenu *menu, TGLay
outHints *l,
                         TGPopupMenu *before = 0);
   virtual TGPopupMenu *AddPopup(const TString &s, Int_t padleft
 = 4, Int_t padright = 0,
                                 Int_t padtop = 0, Int_t padbott
om = 0);
   virtual void AddTitle(TGMenuTitle *title, TGLayoutHints *l, T
GPopupMenu *before = 0);

   virtual TGPopupMenu *GetPopup(const char *s);
   virtual TGPopupMenu *RemovePopup(const char *s);

   virtual TGMenuTitle *GetCurrent() const { return fCurrent; }
   virtual TList  *GetTitles() const { return fTitles; }
   virtual Bool_t  HandleButton(Event_t *event);
   virtual Bool_t  HandleMotion(Event_t *event);
   virtual Bool_t  HandleKey(Event_t *event);
   virtual void    SavePrimitive(std::ostream &out, Option_t *op
tion = "");
   virtual void    Layout();
           void    PopupConnection();
   TGFrameElement* GetLastOnLeft();
```

code

```
#include "TGMenu.h"
```

```
// TGMenuBar
// 最上面那行弹出菜单的标签
TGMenuBar *fMenuBar;// main menu bar
fMenuBar = new TGMenuBar(this, 35, 50, kHorizontalFrame);
fMenuBar->AddPopup("&File", fMenuFile/*子菜单TGPopupMenu*/, new T
GLayoutHints(kLHintsTop|kLHintsLeft, 0, 4, 0, 0));
fMenuBar->AddPopup("&Help", fMenuHelp, new TGLayoutHints(kLHints
Top|kLHintsRight));
AddFrame(fMenuBar, new TGLayoutHints(kLHintsTop | kLHintsExpandX
, 2, 2, 2, 5));
```

```
// TGPopupMenu
// 最上面那行弹出菜单的子菜单
TGPopupMenu *fMenuFile;    // "File" popup menu entry
TGPopupMenu *fMenuHelp;    // "Help" popup menu entry
fMenuFile = new TGPopupMenu(gClient->GetRoot());
fMenuFile->AddEntry(" &Open...\tCtrl+O", IDs, 0,gClient->GetPict
ure("bld_open.png"));
fMenuFile->AddEntry(" &Browse...\tCtrl+B", IDs);
fMenuFile->AddEntry(" &New Canvas\tCtrl+N", IDs);
fMenuFile->AddEntry(" &Close Window\tCtrl+W", IDs);
fMenuFile->AddSeparator();//分割线
fMenuFile->AddEntry(" E&xit\tCtrl+Q", M_FILE_EXIT, 0,gClient->Ge
tPicture("bld_exit.png"));
fMenuFile->Connect("Activated(Int_t)", "DNDMainFrame", this,"Han
dleMenu(Int_t)"/*deal IDs*/);
fMenuFile->DisableEntry(IDs);//显示灰色，无法按
fMenuFile->HideEntry(IDs);//隐藏，不显示
fMenuHelp = new TGPopupMenu(gClient->GetRoot());
fMenuHelp->AddEntry(" &About...", M_HELP_ABOUT, 0,gClient->GetPi
cture("about.xpm"));
fMenuHelp->Connect("Activated(Int_t)", "DNDMainFrame", this,"Han
dleMenu(Int_t)");
fMenuView = new TGPopupMenu(gClient->GetRoot());
fMenuView->AddEntry("&Dock", M_VIEW_DOCK);
fMenuView->DisableEntry(M_VIEW_DOCK);
fMenuView->AddEntry("&Undock", M_VIEW_UNDOCK);
fMenuView->AddSeparator();
fMenuView->AddEntry("Enable U&ndock", M_VIEW_ENBL_DOCK);
fMenuView->AddEntry("Enable &Hide", M_VIEW_ENBL_HIDE);
fMenuView->CheckEntry(M_VIEW_ENBL_DOCK);
fMenuView->CheckEntry(M_VIEW_ENBL_HIDE);
fMenuDock->Connect("Undocked()", "TestMainFrame", this, "HandleM
enu(=M_VIEW_UNDOCK)");
```

example

TGNumberEntry

TGNumberEntryField 继承 TGTextEntry, TGNumberFormat

TGNumberEntry 继承 TGCompositeFrame, TGWidget, TGNumberFormat

class

TGNumberFormat

```
class TGNumberFormat {
public:
   enum EStyle {              // Style of number entry field
      kNESInteger = 0,        // Integer
      kNESRealOne = 1,        // Fixed fraction real, one digit
      kNESRealTwo = 2,        // Fixed fraction real, two digit
      kNESRealThree = 3,      // Fixed fraction real, three digit
      kNESRealFour = 4,       // Fixed fraction real, four digit
      kNESReal = 5,           // Real number
      kNESDegree = 6,         // Degree
      kNESMinSec = 7,         // Minute:seconds
      kNESHourMin = 8,        // Hour:minutes
      kNESHourMinSec = 9,     // Hour:minute:seconds
      kNESDayMYear = 10,      // Day/month/year
      kNESMDayYear = 11,      // Month/day/year
      kNESHex = 12            // Hex
   };

   enum EAttribute {          // Attributes of number entry field
      kNEAAnyNumber = 0,      // Any number
      kNEANonNegative = 1,    // Non-negative number
      kNEAPositive = 2        // Positive number
   };

   enum ELimit {              // Limit selection of number entry
field
      kNELNoLimits = 0,       // No limits
```

```
      kNELLimitMin = 1,       // Lower limit only
      kNELLimitMax = 2,       // Upper limit only
      kNELLimitMinMax = 3     // Both lower and upper limits
   };

   enum EStepSize {           // Step for number entry field incr
ease
      kNSSSmall = 0,          // Small step
      kNSSMedium = 1,         // Medium step
      kNSSLarge = 2,          // Large step
      kNSSHuge = 3            // Huge step
   };

   virtual ~TGNumberFormat() { }
   ClassDef(TGNumberFormat,0)  // Class defining namespace for s
everal enums used by TGNumberEntry
};
```

TGNumberEntryField

```
   TGNumberEntryField(const TGWindow *p, Int_t id,
                      Double_t val, GContext_t norm,
                      FontStruct_t font = GetDefaultFontStruct()
,
                      UInt_t option = kSunkenFrame | kDoubleBord
er,
                      Pixel_t back = GetWhitePixel());
   TGNumberEntryField(const TGWindow *parent = 0,
                      Int_t id = -1, Double_t val = 0,
                      EStyle style = kNESReal,
                      EAttribute attr = kNEAAnyNumber,
                      ELimit limits = kNELNoLimits,
                      Double_t min = 0, Double_t max = 1);

   virtual void SetNumber(Double_t val);
   virtual void SetIntNumber(Long_t val);
   virtual void SetTime(Int_t hour, Int_t min, Int_t sec);
   virtual void SetDate(Int_t year, Int_t month, Int_t day);
   virtual void SetHexNumber(ULong_t val);
```

```
   virtual void SetText(const char* text, Bool_t emit = kTRUE);

   virtual Double_t GetNumber() const;
   virtual Long_t   GetIntNumber() const;
   virtual void     GetTime(Int_t& hour, Int_t& min, Int_t& sec)
const;
   virtual void     GetDate(Int_t& year, Int_t& month, Int_t& da
y) const;
   virtual ULong_t  GetHexNumber() const;

   virtual Int_t GetCharWidth(const char* text = "0") const;
   virtual void  IincreaseNumber(EStepSize step = kNSSSmall,
                                 Int_t sign = 1, Bool_t logstep =
 kFALSE);
   virtual void  SetFormat(EStyle style,
                           EAttribute attr = kNEAAnyNumber);
   virtual void  SetLimits(ELimit limits = kNELNoLimits,
                           Double_t min = 0, Double_t max = 1);
   virtual void  SetState(Bool_t state);
   virtual void  SetLogStep(Bool_t on = kTRUE) {
      // Set logarithmic steps
      fStepLog = on; }

   virtual EStyle GetNumStyle() const {
      // Get the numerical style
      return fNumStyle; }
   virtual EAttribute GetNumAttr() const {
      // Get the numerical attribute
      return fNumAttr; }
   virtual ELimit GetNumLimits() const {
      // Get the numerialc limit attribute
      return fNumLimits; }
   virtual Double_t GetNumMin() const {
      // Get the lower limit
      return fNumMin; }
   virtual Double_t GetNumMax() const {
      // Get the upper limit
      return fNumMax; }
   virtual Bool_t IsLogStep() const {
      // Is log step enabled?
```

```
      return fStepLog; }

   virtual Bool_t HandleKey(Event_t* event);
   virtual Bool_t HandleFocusChange (Event_t* event);
   virtual void   TextChanged(const char *text = 0);
   virtual void   ReturnPressed();
   virtual void   Layout();
   virtual Bool_t IsEditable() const { return kFALSE; }
   virtual void   InvalidInput(const char *instr) { Emit("Invali
dInput(char*)", instr); }   //*SIGNAL*
   virtual void   SavePrimitive(std::ostream &out, Option_t * =
"");
```

TGNumberEntry

```
   TGNumberEntry(const TGWindow *parent = 0, Double_t val = 0,
                 Int_t digitwidth = 5, Int_t id = -1,
                 EStyle style = kNESReal,
                 EAttribute attr = kNEAAnyNumber,
                 ELimit limits = kNELNoLimits,
                 Double_t min = 0, Double_t max = 1);
   virtual ~TGNumberEntry();

   virtual void SetNumber(Double_t val) {
      // Set the numeric value (floating point representation)
      fNumericEntry->SetNumber(val); }
   virtual void SetIntNumber(Long_t val) {
      // Set the numeric value (integer representation)
      fNumericEntry->SetIntNumber(val); }
   virtual void SetTime(Int_t hour, Int_t min, Int_t sec) {
      // Set the numeric value (time format)
      fNumericEntry->SetTime(hour, min, sec); }
   virtual void SetDate(Int_t year, Int_t month, Int_t day) {
      // Set the numeric value (date format)
      fNumericEntry->SetDate(year, month, day); }
   virtual void SetHexNumber(ULong_t val) {
      // Set the numeric value (hex format)
      fNumericEntry->SetHexNumber(val); }
   virtual void SetText(const char* text) {
```

```
      // Set the value (text format)
      fNumericEntry->SetText(text); }
   virtual void SetState(Bool_t enable = kTRUE);

   virtual Double_t GetNumber() const {
      // Get the numeric value (floating point representation)
      return fNumericEntry->GetNumber(); }
   virtual Long_t GetIntNumber() const {
      // Get the numeric value (integer representation)
      return fNumericEntry->GetIntNumber (); }
   virtual void GetTime(Int_t& hour, Int_t& min, Int_t& sec) con
st {
      // Get the numeric value (time format)
      fNumericEntry->GetTime(hour, min, sec); }
   virtual void GetDate(Int_t& year, Int_t& month, Int_t& day) c
onst {
      // Get the numeric value (date format)
      fNumericEntry->GetDate(year, month, day); }
   virtual ULong_t GetHexNumber() const {
      // Get the numeric value (hex format)
      return fNumericEntry->GetHexNumber(); }
   virtual void IncreaseNumber(EStepSize step = kNSSSmall,
                               Int_t sign = 1, Bool_t logstep =
kFALSE) {
      // Increase the number value
      fNumericEntry->IncreaseNumber(step, sign, logstep); }
   virtual void SetFormat(EStyle style, EAttribute attr = TGNumb
erFormat::kNEAAnyNumber) {
      // Set the numerical format
      fNumericEntry->SetFormat(style, attr); }
   virtual void SetLimits(ELimit limits = TGNumberFormat::kNELNo
Limits,
                          Double_t min = 0, Double_t max = 1) {
      // Set the numerical limits.
      fNumericEntry->SetLimits(limits, min, max); }

   virtual EStyle GetNumStyle() const {
      // Get the numerical style
      return fNumericEntry->GetNumStyle(); }
   virtual EAttribute GetNumAttr() const {
```

```
      // Get the numerical attribute
      return fNumericEntry->GetNumAttr(); }
   virtual ELimit GetNumLimits() const {
      // Get the numerical limit attribute
      return fNumericEntry->GetNumLimits(); }
   virtual Double_t GetNumMin() const {
      // Get the lower limit
      return fNumericEntry->GetNumMin(); }
   virtual Double_t GetNumMax() const {
      // Get the upper limit
      return fNumericEntry->GetNumMax(); }
   virtual Bool_t IsLogStep() const {
      // Is log step enabled?
      return fNumericEntry->IsLogStep(); }
   virtual void   SetButtonToNum(Bool_t state);

   void SetNumStyle(EStyle style) {
         SetFormat(style, GetNumAttr()); }                  //*S
UBMENU*
   void SetNumAttr(EAttribute attr = kNEAAnyNumber) {
         SetFormat(GetNumStyle(), attr); }                  //*S
UBMENU*
   void SetNumLimits(ELimit limits = kNELNoLimits) {
         SetLimits(limits, GetNumMin(), GetNumMax());  }    //*S
UBMENU*
   void SetLimitValues(Double_t min = 0, Double_t max = 1) {
         SetLimits(GetNumLimits(), min, max);  }            //*M
ENU*
   virtual void SetLogStep(Bool_t on = kTRUE);              //*T
OGGLE* *GETTER=IsLogStep

   virtual void   Associate(const TGWindow *w);
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);
   virtual void   ValueChanged(Long_t val);     //*SIGNAL*
   virtual void   ValueSet(Long_t val);         //*SIGNAL*

   TGNumberEntryField *GetNumberEntry() const {
      // Get the number entry field
      return fNumericEntry; }
```

```
   TGButton *GetButtonUp() const {
      // Get the up button
      return fButtonUp; }
   TGButton *GetButtonDown() const {
      // Get the down button
      return fButtonDown; }

   virtual Bool_t IsEditable() const { return kFALSE; }

   UInt_t GetDefaultHeight() const { return fNumericEntry->GetDe
faultHeight(); }
   virtual void SavePrimitive(std::ostream &out, Option_t * = "")
;
   virtual TGLayoutManager *GetLayoutManager() const;
```

code

```
#include "TGNumberEntry.h"

// TGNumberEntry
TGNumberEntry *fYearEntry = new TGNumberEntry(frame, 2016/*初始值
*/, 5/*能容纳位数*/, IDs,TGNumberFormat::kNESInteger,TGNumberForma
t::kNEAPositive,TGNumberFormat::kNELLimitMin, 1995/*min*/,2020/*
max*/);
frame->AddFrame(fYearEntry, new TGLayoutHints(kLHintsLeft, 5, 5,
2, 2));
// fYearEntry->Connect("ValueSet(Long_t)", "MyMainFrame", this,
"DoSetlabel()");
// (fYearEntry->GetNumberEntry())->Connect("ReturnPressed()", "M
yMainFrame", this,"DoSetlabel()");
```

example

TGObject

This class is the baseclass for all ROOT GUI widgets.
The ROOT GUI components emulate the Win95 look and feel and the code is based on the XClass'95 code (see Copyleft in source).

TGObject 继承 TObject

class

```
   TGObject(): fId(0), fClient(0) { }
   TGObject(const TGObject& tgo): TObject(tgo), fId(tgo.fId), fC
lient(tgo.fClient) { }
   virtual ~TGObject() { }

   Handle_t  GetId() const { return fId; }
   TGClient *GetClient() const { return fClient; }
   ULong_t   Hash() const { return (ULong_t) fId >> 0; }
   Bool_t    IsEqual(const TObject *obj) const;
/// Equal comparison (TGObjects are equal if they have the same
/// window identifier). If the TGObjects have not been created by

/// the Window manager (e.g. a TGLVEntry), then fall back to the
/// default TObject equal comparison

   virtual void SaveAs(const char* filename = "", Option_t* opti
on = "") const;
/// Write this TGObject to a file using TImage, if filename's ex
tension signals
/// a valid TImage::EImageFileType, as defined by TImage::GetIma
geFileTypeFromFilename().
/// Otherwise forward to TObject::SaveAs().
```

code

example

TGPicture

TGPicture 继承 TObject, TRefCnt , friend TGPicturePool

TGSelectedPicture 继承 TGPicture

TGPicturePool 继承 TObject

class

TGPicture

```
   virtual ~TGPicture();

   const char *GetName() const { return fName; }
   UInt_t      GetWidth() const { return fAttributes.fWidth; }
   UInt_t      GetHeight() const { return fAttributes.fHeight; }
   Pixmap_t    GetPicture() const { return fPic; }
   Pixmap_t    GetMask() const { return fMask; }
   Bool_t      IsScaled() const { return fScaled; }
   ULong_t     Hash() const { return fName.Hash(); }
   static const char *HashName(const char *name, Int_t width, In
t_t height);

   virtual void Draw(Handle_t id, GContext_t gc, Int_t x, Int_t
y) const;
   void         Print(Option_t *option="") const;
```

TGSelectedPicture

```
   TGSelectedPicture(const TGClient *client, const TGPicture *p)
;
   virtual ~TGSelectedPicture();
```

TGPicturePool

```
   TGPicturePool(const TGClient *client, const char *path):
      fClient(client), fPath(path), fPicList(0) { }
   virtual ~TGPicturePool();

   const char      *GetPath() const { return fPath; }
   const TGPicture *GetPicture(const char *name);
   const TGPicture *GetPicture(const char *name, char **xpm);
   const TGPicture *GetPicture(const char *name, UInt_t new_widt
h, UInt_t new_height);
   const TGPicture *GetPicture(const char *name, Pixmap_t pxmap,
 Pixmap_t mask =  0);
   void             FreePicture(const TGPicture *pic);

   void             Print(Option_t *option="") const;
```

code

```
#include "TGIcon.h"
#include "TGResourcePool.h"
#include "TGPicture.h"

// TGPicture TGPicturePool TGIcon
// 插入系统自带的小图片
const char * const icon1[] =
{
"16 16 8 1",
"    c None s None",
".   c #808080",
"X   c #FFFF00",
"o   c #c0c0c0",
"O   c black",
"+   c #00FFFF",
"@   c #00FF00",
"#   c white",
"      .....      ",
"   ..XXoooOO    ",
"  .+XXXoooooO   ",
" .@++XXoooo#oO  ",
```

```
" .@@+XXooo#ooO  ",
".oo@@+Xoo#oooo0 ",
".ooo@+.O.oooooO ",
".oooo@O#Ooooooo ",
".oooo#.O.+oooo0 ",
".ooo#oo#@X+ooo0 ",
" .o#oooo@X++oO  ",
" .#ooooo@XX++O  ",
"  .ooooo@@XXO   ",
"   ..ooo@@OO    ",
"     ..OOO      ",
"                "
};
TString name = "myicon";
ULong_t yellow;
gClient->GetColorByName("yellow", yellow);
TGPicturePool *picpool = gClient->GetResourcePool()->GetPictureP
ool();
const TGPicture *iconpic = picpool->GetPicture(name.Data(),(char
 **)icon1);
TGIcon *icon = new TGIcon(frame, iconpic, 40, 40, kChildFrame, y
ellow);
frame->AddFrame(icon, new TGLayoutHints(kLHintsLeft, 1,15,1,1));
```

example

TGProgressBar

TGProgressBar, TGHProgressBar and TGVProgressBar

The classes in this file implement progress bars. Progress bars can be used to show progress of tasks taking more then a few seconds. TGProgressBar is an abstract base class, use either TGHProgressBar or TGVProgressBar. TGHProgressBar can in addition show the position as text in the bar.

TGProgressBar 继承 TGFrame

TGHProgressBar 继承 TGProgressBar

TGVProgressBar 继承 TGProgressBar

class

TGProgressBar

```
   enum EBarType { kStandard, kFancy };
   enum EFillType { kSolidFill, kBlockFill };
   enum { kProgressBarStandardWidth = 16, kProgressBarTextWidth
= 24,
          kBlockSize = 8, kBlockSpace = 2 };

   static FontStruct_t  GetDefaultFontStruct();/// Return defaul
t font structure in use.
   static const TGGC   &GetDefaultGC();/// Return default graphi
cs context in use.

   TGProgressBar(const TGWindow *p, UInt_t w, UInt_t h,
                 Pixel_t back = GetWhitePixel(),
                 Pixel_t barcolor = GetDefaultSelectedBackground
(),
                 GContext_t norm = GetDefaultGC()(),
                 FontStruct_t font = GetDefaultFontStruct(),
                 UInt_t options = kDoubleBorder | kSunkenFrame);
```

```
/// Create progress bar.
   virtual ~TGProgressBar() { }

   Float_t      GetMin() const { return fMin; }
   Float_t      GetMax() const { return fMax; }
   Float_t      GetPosition() const { return fPos; }
   EFillType    GetFillType() const { return fFillType; }
   EBarType     GetBarType() const { return fBarType; }
   Bool_t       GetShowPos() const { return fShowPos; }
   TString      GetFormat() const { return fFormat; }
   const char*  GetValueFormat() const { return fFormat.Data();
}
   Bool_t       UsePercent() const { return fPercent; }
   Pixel_t      GetBarColor() const { return fBarColorGC.GetFore
ground(); }
   GContext_t   GetNormGC() const { return fNormGC; }
   FontStruct_t GetFontStruct() const { return fFontStruct; }

   void         SetPosition(Float_t pos);                //*MENU
*  *GETTER=GetPosition
/// Set progress position between [min,max].

   void         SetRange(Float_t min, Float_t max);      //*MENU*

/// Set min and max of progress bar.

   void         Increment(Float_t inc);/// Increment progress po
sition.
   void         SetBarType(EBarType type);               //*SUBM
ENU*
/// Set bar type.

   void         SetFillType(EFillType type);             //*SUBM
ENU*
/// Set fill type.

   virtual void Percent(Bool_t on) { fPercent = on; fClient->Nee
dRedraw(this); } //*TOGGLE* *GETTER=UsePercent
   virtual void ShowPos(Bool_t on) { fShowPos = on; fClient->Nee
dRedraw(this); } //*TOGGLE* *GETTER=GetShowPos
```

```
   virtual void Format(const char *format = "%.2f");     //*MENU
* *GETTER=GetValueFormat
/// Set format for displaying a value.

   void         SetMin(Float_t min) { fMin = min; }
   void         SetMax(Float_t max) { fMax = max; }
   virtual void SetBarColor(Pixel_t color);/// Set progress bar
color.
   void         SetBarColor(const char *color="blue");/// Set pr
ogress bar color.
   virtual void Reset();                                 //*MENU*

/// Reset progress bar (i.e. set pos to 0).

   virtual void SetForegroundColor(Pixel_t pixel);
/// Change text color drawing.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save progress bar parameters as a C++ statement(s) on output
 stream out.
```

TGHProgressBar

```
   TGHProgressBar(const TGWindow *p = 0,
                  UInt_t w = 4, UInt_t h = kProgressBarTextWidth
,
                  Pixel_t back = GetWhitePixel(),
                  Pixel_t barcolor = GetDefaultSelectedBackgroun
d(),
                  GContext_t norm = GetDefaultGC()(),
                  FontStruct_t font = GetDefaultFontStruct(),
                  UInt_t options = kDoubleBorder | kSunkenFrame);
/// Horizontal progress bar constructor.
   TGHProgressBar(const TGWindow *p, EBarType type, UInt_t w);
/// Simple constructor allow you to create either a standard pro
gress
/// bar, or a more fancy progress bar (fancy means: double sized
 border,
/// white background and a bit wider to allow for text to be pri
nted
/// in the bar.

   virtual ~TGHProgressBar() { }

   virtual TGDimension GetDefaultSize() const
                     { return TGDimension(fWidth, fBarWidth); }

   void ShowPosition(Bool_t set = kTRUE, Bool_t percent = kTRUE,
                     const char *format = "%.2f");
/// Show postion text, either in percent or formatted according
format.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a horizontal progress bar as a C++ statement(s) on outp
ut stream out
```

TGVProgressBar

```
   TGVProgressBar(const TGWindow *p = 0,
                  UInt_t w = kProgressBarTextWidth, UInt_t h = 4
,
                  Pixel_t back = GetWhitePixel(),
                  Pixel_t barcolor = GetDefaultSelectedBackgroun
d(),
                  GContext_t norm = GetDefaultGC()(),
                  FontStruct_t font = GetDefaultFontStruct(),
                  UInt_t options = kDoubleBorder | kSunkenFrame);
/// cconstructor
   TGVProgressBar(const TGWindow *p, EBarType type, UInt_t h);
/// Simple constructor allow you to create either a standard pro
gress
/// bar, or a more fancy progress bar (fancy means: double sized
 border,
/// white background and a bit wider to allow for text to be pri
nted
/// in the bar.

   virtual ~TGVProgressBar() { }

   virtual TGDimension GetDefaultSize() const
                     { return TGDimension(fBarWidth, fHeight); }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
   void ShowPos(Bool_t) { }
   void Percent(Bool_t) { }
/// Save a vertical progress bar as a C++ statement(s) on output
 stream out.
```

code

example

TGScrollBar

The classes in this file implement scrollbars. Scrollbars can be either placed horizontal or vertical. A scrollbar contains three TGScrollBarElements: The "head", "tail" and "slider". The head and tail are fixed at either end and have the typical arrows in them.

```
The TGHScrollBar will generate the following event messages:

kC_HSCROLL, kSB_SLIDERPOS, position, 0

kC_HSCROLL, kSB_SLIDERTRACK, position, 0


The TGVScrollBar will generate the following event messages:

kC_VSCROLL, kSB_SLIDERPOS, position, 0

kC_VSCROLL, kSB_SLIDERTRACK, position, 0
```

TGScrollBarElement 继承 TGFrame

TGScrollBar 继承 TGFrame, TGWidget

TGHScrollBar 继承 TGScrollBar

TGVScrollBar 继承 TGScrollBar

class

```
//--- scrollbar types

enum EScrollBarMode {
   kSBHorizontal,
   kSBVertical
};
```

TGScrollBarElement

```
   TGScrollBarElement(const TGWindow *p = 0, const TGPicture *pi
c = 0,
                      UInt_t w = 1, UInt_t h = 1,
                      UInt_t options = kRaisedFrame | kDoubleBor
der,
                      Pixel_t back = GetDefaultFrameBackground()
);
   virtual ~TGScrollBarElement();

   virtual void SetState(Int_t state);
   virtual void DrawBorder();
   virtual void SetEnabled(Bool_t on = kTRUE);
   virtual Bool_t IsEnabled() const { return !(fState & kButtonD
isabled); }
   virtual Bool_t HandleCrossing(Event_t *event);
```

TGScrollBar

```
   static Pixmap_t  GetBckgndPixmap();
   static Int_t     GetScrollBarWidth();

   TGScrollBar(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1
,
               UInt_t options = kChildFrame,
               Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGScrollBar();

   void           GrabPointer(Bool_t grab) { fGrabPointer = grab
; }

   virtual void   DrawBorder() { }
   virtual Bool_t HandleButton(Event_t *event) = 0;
   virtual Bool_t HandleCrossing(Event_t *event);
   virtual Bool_t HandleMotion(Event_t *event) = 0;
   virtual Bool_t HandleTimer(TTimer *t);
   virtual void   Layout() = 0;
```

```
   virtual void  SetDragging(Bool_t drag) { fDragging = drag; }
   virtual void  SetRange(Int_t range, Int_t page_size) = 0;
   virtual void  SetPosition(Int_t pos) = 0;
   virtual Int_t GetPosition() const { return fPos; }
   virtual Int_t GetPageSize() const { return fPsize; }
   virtual Int_t GetRange() const { return fRange; }
   virtual void  Resize(UInt_t w = 0, UInt_t h = 0) { TGFrame::R
esize(w, h); SetRange(fRange, fPsize); }
   virtual void  MoveResize(Int_t x, Int_t y, UInt_t w = 0, UInt
_t h = 0)
                  { TGFrame::MoveResize(x, y, w, h); SetRange(fR
ange, fPsize); }
   virtual void  Resize(TGDimension size) { Resize(size.fWidth,
size.fHeight); }
   virtual void  ChangeBackground(Pixel_t back);
   virtual void  SetAccelerated(Bool_t m = kTRUE) { fAccelerated
 = m; }
         Bool_t  IsAccelerated() const { return fAccelerated; }

   virtual void MapSubwindows() { TGWindow::MapSubwindows(); }
   TGScrollBarElement *GetHead() const { return fHead; }
   TGScrollBarElement *GetTail() const { return fTail; }
   TGScrollBarElement *GetSlider() const { return fSlider; }

   virtual void  PositionChanged(Int_t pos) { Emit("PositionChan
ged(Int_t)", pos); } //*SIGNAL*
   virtual void  RangeChanged(Int_t range) { Emit("RangeChanged(
Int_t)", range); } //*SIGNAL*
   virtual void  PageSizeChanged(Int_t range) { Emit("PageSizeCh
anged(Int_t)", range); } //*SIGNAL*

   virtual Int_t GetSmallIncrement() { return fSmallInc; }
   virtual void  SetSmallIncrement(Int_t increment) { fSmallInc
= increment; }
```

TGHScrollBar

```
   TGHScrollBar(const TGWindow *p = 0, UInt_t w = 4, UInt_t h = 2
,
                UInt_t options = kHorizontalFrame,
                Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGHScrollBar() { }

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleMotion(Event_t *event);
   virtual TGDimension GetDefaultSize() const
                        { return TGDimension(fWidth, GetScrollBa
rWidth()); }
   virtual void Layout();

   virtual void SetRange(Int_t range, Int_t page_size);  //*MENU*

   virtual void SetPosition(Int_t pos);                  //*MENU
* *GETTER=GetPosition
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

TGVScrollBar

```
   TGVScrollBar(const TGWindow *p = 0, UInt_t w = 2, UInt_t h = 4
,
                UInt_t options = kVerticalFrame,
                Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGVScrollBar() { }

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleMotion(Event_t *event);
   virtual TGDimension GetDefaultSize() const
                        { return TGDimension(GetScrollBarWidth()
, fHeight); }
   virtual void Layout();

   virtual void SetRange(Int_t range, Int_t page_size);  //*MENU*

   virtual void SetPosition(Int_t pos);                  //*MENU
*  *GETTER=GetPosition
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

code

example

TGSimpleTable

TGSimpleTable 继承 TGTable

A simple table that owns it's interface.

class

```
TGSimpleTable(TGWindow *p, Int_t id, Double_t **data,
              UInt_t nrows, UInt_t ncolumns);
virtual ~TGSimpleTable();
```

code

```
#include "TGSimpleTable.h"

// TGSimpleTable
// 简单数据表格
Double_t     **fData;
UInt_t         fNDataRows;
UInt_t         fNDataColumns;
UInt_t         fNTableRows;
UInt_t         fNTableColumns;
TGSimpleTable *fSimpleTable;
Int_t i = 0, j = 0;
fData = new Double_t*[fNDataRows];// Create the needed data.
for (i = 0; i < (Int_t)fNDataRows; i++) {
   fData[i] = new Double_t[fNDataColumns];
   for (j = 0; j < (Int_t)fNDataColumns; j++) {
      fData[i][j] = 10 * i + j;
   }
}
fSimpleTable = new TGSimpleTable(frame, IDs, fData, fNTableRows,
 fNTableColumns);
frame->AddFrame(fSimpleTable, new TGLayoutHints(kLHintsExpandX |
 kLHintsExpandY));
```

example

TGSlider

Slider widgets allow easy selection of a range.
Sliders can be either horizontal or vertical oriented and there is
a choice of two different slider types and three different types
of tick marks.

TGSlider is an abstract base class. Use the concrete TGVSlider and TGHSlider.

```
Dragging the slider will generate the event:

kC_VSLIDER, kSL_POS, slider id, position  (for vertical slider)

kC_HSLIDER, kSL_POS, slider id, position  (for horizontal slider
)

Pressing the mouse will generate the event:

kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)


Releasing the mouse will generate the event:

kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider)
```

TGSlider 继承 TGFrame, TGWidget

TGVSlider 继承 TGSlider
Vertical slider widget

TGHSlider 继承 TGSlider
Horizontal slider widget

class

```
//--- sizes for vert. and horz. sliders

enum ESliderSize {
   kSliderWidth  = 24,
   kSliderHeight = kSliderWidth
};


enum ESliderType {
   //--- slider types (type of slider picture)
   kSlider1        = BIT(0),
   kSlider2        = BIT(1),

   //--- scaling of slider
   kScaleNo        = BIT(2),
   kScaleDownRight = BIT(3),
   kScaleBoth      = BIT(4)
};
```

TGSlider

```
   TGSlider(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
            UInt_t type = kSlider1 | kScaleBoth, Int_t id = -1,
            UInt_t options = kChildFrame,
            Pixel_t back = GetDefaultFrameBackground());

   virtual ~TGSlider() { }

   virtual Bool_t HandleButton(Event_t *event) = 0;
   virtual Bool_t HandleConfigureNotify(Event_t* event) = 0;
   virtual Bool_t HandleMotion(Event_t *event) = 0;

   virtual void  SetEnabled(Bool_t flag = kTRUE) { SetState( fla
g ); }              //*TOGGLE* *GETTER=IsEnabled
   virtual void  SetState(Bool_t state);
   virtual void  SetScale(Int_t scale) { fScale = scale; }
                  //*MENU*
   virtual void  SetRange(Int_t min, Int_t max) { fVmin = min; f
Vmax = max; }       //*MENU*
   virtual void  SetPosition(Int_t pos) { fPos = pos; fClient->N
eedRedraw(this); }  //*MENU*
   virtual Int_t GetPosition() const { return fPos; }
   virtual Int_t GetMinPosition() const { return fVmin; }
   virtual Int_t GetMaxPosition() const { return fVmax; }
   virtual Int_t GetScale() const { return fScale; }
   virtual void  MapSubwindows() { TGWindow::MapSubwindows(); }
   virtual void  ChangeSliderPic(const char *name) {
                  if (fSliderPic) fClient->FreePicture(fSlider
Pic);
                  fSliderPic = fClient->GetPicture(name);
                }

   virtual void  PositionChanged(Int_t pos) { Emit("PositionChan
ged(Int_t)", pos); } // *SIGNAL*
   virtual void  Pressed() { Emit("Pressed()"); }    // *SIGNAL*
   virtual void  Released() { Emit("Released()"); }  // *SIGNAL*
```

TGVSlider

```
   TGVSlider(const TGWindow *p = 0, UInt_t h = 40,
             UInt_t type = kSlider1 | kScaleBoth, Int_t id = -1,
             UInt_t options = kVerticalFrame,
             Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGVSlider();

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleConfigureNotify(Event_t* event);
   virtual Bool_t HandleMotion(Event_t *event);
   virtual TGDimension GetDefaultSize() const
                     { return TGDimension(kSliderWidth, fHeight)
; }
   virtual void   Resize(UInt_t w, UInt_t h) { TGFrame::Resize(w
, h ? h+16 : fHeight + 16); }
   virtual void   Resize(TGDimension size) { Resize(size.fWidth,
 size.fHeight); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
```

TGHSlider

```
   TGHSlider(const TGWindow *p = 0, UInt_t w = 40,
             UInt_t type = kSlider1 | kScaleBoth, Int_t id = -1,
             UInt_t options = kHorizontalFrame,
             Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGHSlider();

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleConfigureNotify(Event_t* event);
   virtual Bool_t HandleMotion(Event_t *event);
   virtual TGDimension GetDefaultSize() const
                     { return TGDimension(fWidth, kSliderHeight)
; }
   virtual void   Resize(UInt_t w, UInt_t h) { TGFrame::Resize(w
 ? w+16 : fWidth + 16, h); }
   virtual void   Resize(TGDimension size) { Resize(size.fWidth,
 size.fHeight); }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
```

code

example

TGSpeedo

TGSpeedo is a widget looking like a speedometer, with a needle, a counter and a small odometer window.

Three thresholds are configurable, with their glowing color
A peak mark can be enabled, allowing to keep track of the highest value displayed. The mark can be reset by right-clicking on the widget.
Two signals are available:

- OdoClicked(): when user click on the small odometer window
- LedClicked(): when user click on the small led near the counter

TGSpeedo 继承 TGFrame, TGWidget

class

```
   enum EGlowColor { kNoglow, kGreen, kOrange, kRed };

   TGSpeedo(const TGWindow *p = 0, int id = -1);/// TGSpeedo wid
get constructor.
   TGSpeedo(const TGWindow *p, Float_t smin, Float_t smax,
            const char *lbl1 = "", const char *lbl2 = "",
            const char *dsp1 = "", const char *dsp2 = "", int id
 = -1);/// TGSpeedo widget constructor.
   virtual ~TGSpeedo();/// TGSpeedo widget Destructor.

   virtual TGDimension  GetDefaultSize() const;/// Return defaul
t dimension of the widget.
   virtual Bool_t       HandleButton(Event_t *event);/// Handle
mouse button event.

   const TGPicture     *GetPicture() const { return fBase; }
   TImage              *GetImage() const { return fImage; }
   Float_t              GetPeakVal() const { return fPeakVal; }
   Float_t              GetScaleMin() const { return fScaleMin;
```

```
}
   Float_t              GetScaleMax() const { return fScaleMax;
}
   Bool_t               IsThresholdActive() { return fThresholdA
ctive; }

   void Build();/// Build TGSpeedo widget.
   void Glow(EGlowColor col = kGreen);/// Make speedo glowing.
   void StepScale(Float_t step);/// Increment/decrement scale (n
eedle position) of "step" value.
   void SetScaleValue(Float_t val);/// Set actual scale (needle
position) value.
   void SetScaleValue(Float_t val, Int_t damping);/// Set actual
 scale (needle position) value.
   void SetOdoValue(Int_t val);/// Set actual value of odo meter.

   void SetDisplayText(const char *text1, const char *text2 = "")
;
/// Set small display text (two lines).

   void SetLabelText(const char *text1, const char *text2 = "");
/// Set main label text (two lines).

   void SetMinMaxScale(Float_t min, Float_t max);/// Set min and
 max scale values.
   void SetThresholds(Float_t th1 = 0.0, Float_t th2 = 0.0, Floa
t_t th3 = 0.0)
             { fThreshold[0] = th1; fThreshold[1] = th2; fThresh
old[2] = th3; }
   void SetThresholdColors(EGlowColor col1, EGlowColor col2, EGl
owColor col3)
             { fThresholdColor[0] = col1; fThresholdColor[1] = c
ol2; fThresholdColor[2] = col3; }
   void EnableThreshold() { fThresholdActive = kTRUE; }
   void DisableThreshold() { fThresholdActive = kFALSE; Glow(kNo
glow); fClient->NeedRedraw(this);}
   void EnablePeakMark() { fPeakMark = kTRUE; }
   void DisablePeakMark() { fPeakMark = kFALSE; }
   void EnableMeanMark() { fMeanMark = kTRUE; }
   void DisableMeanMark() { fMeanMark = kFALSE; }
```

```
   void ResetPeakVal() { fPeakVal = fValue; fClient->NeedRedraw(
this); }
   void SetMeanValue(Float_t mean) { fMeanVal = mean; fClient->N
eedRedraw(this); }

   void OdoClicked() { Emit("OdoClicked()"); }   // *SIGNAL*
   void LedClicked() { Emit("LedClicked()"); }   // *SIGNAL*
```

code

example

TGSplitter

TGSplitter, TGVSplitter and TGHSplitter

A splitter allows the frames left and right or above and below of it to be resized. The frame to be resized must have the kFixedWidth or kFixedHeight property set.

TGSplitter 继承 TGFrame

TGVSplitter 继承 TGSplitter

TGHSplitter 继承 TGSplitter

TGVFileSplitter 继承 TGVSplitter

class

TGSplitter

```
   TGSplitter(const TGWindow *p = 0, UInt_t w = 2, UInt_t h = 4,
              UInt_t options = kChildFrame,
              Pixel_t back = GetDefaultFrameBackground());/// Cr
eate a splitter.
   virtual ~TGSplitter() { }

   virtual void   SetFrame(TGFrame *frame, Bool_t prev) = 0;

   virtual Bool_t HandleButton(Event_t *event) = 0;
   virtual Bool_t HandleMotion(Event_t *event) = 0;
   virtual Bool_t HandleCrossing(Event_t *event) = 0;

   void DragStarted();      // *SIGNAL*  /// Emit DragStarted si
gnal.
   void Moved(Int_t delta); // *SIGNAL*  /// Emit Moved signal.

   Bool_t GetExternalHandler() const { return fExternalHandler;
}
   void SetExternalHandler(Bool_t x) { fExternalHandler = x; }
```

TGVSplitter

```
   TGVSplitter(const TGWindow *p = 0, UInt_t w = 4, UInt_t h = 4
,
               UInt_t options = kChildFrame,
               Pixel_t back = GetDefaultFrameBackground());/// C
reate a vertical splitter.
   TGVSplitter(const TGWindow *p, UInt_t w, UInt_t h, Bool_t ext
ernal);/// Create a vertical splitter.
   virtual ~TGVSplitter();/// Delete vertical splitter widget.

   virtual void   DrawBorder();/// Draw vertical splitter.
   virtual void   SetFrame(TGFrame *frame, Bool_t left);
/// Set frame to be resized. If frame is on the left of the spli
tter
/// set left to true.

   const TGFrame *GetFrame() const { return fFrame; }
   Bool_t         GetLeft() const { return fLeft; }
   Bool_t         IsLeft() const { return fLeft; }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a splitter widget as a C++ statement(s) on output strea
m out.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in vertical splitter.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion event in vertical splitter.
   virtual Bool_t HandleCrossing(Event_t *event);/// Handle mous
e motion event in vertical splitter.
```

TGHSplitter

```
   TGHSplitter(const TGWindow *p = 0, UInt_t w = 4, UInt_t h = 4
,
               UInt_t options = kChildFrame,
               Pixel_t back = GetDefaultFrameBackground());/// C
reate a horizontal splitter.
   TGHSplitter(const TGWindow *p, UInt_t w, UInt_t h, Bool_t ext
ernal);/// Create a horizontal splitter.
   virtual ~TGHSplitter();/// Delete horizontal splitter widget.

   virtual void   DrawBorder();/// Draw horizontal splitter.
   virtual void   SetFrame(TGFrame *frame, Bool_t above);
/// Set frame to be resized. If frame is above the splitter
/// set above to true.

   const TGFrame *GetFrame() const { return fFrame; }
   Bool_t         GetAbove() const { return fAbove; }
   Bool_t         IsAbove() const { return fAbove; }
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a splitter widget as a C++ statement(s) on output strea
m out.

   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in horizontal splitter.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion event in horizontal splitter.
   virtual Bool_t HandleCrossing(Event_t *event);/// Handle mous
e motion event in horizontal splitter.
```

TGVFileSplitter

```
   TGVFileSplitter(const TGWindow *p = 0, UInt_t w = 4, UInt_t h
 = 4,
                   UInt_t options = kChildFrame,
                   Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGVFileSplitter();

   virtual Bool_t HandleDoubleClick(Event_t *);/// Handle double
 click mouse event in splitter.
   virtual Bool_t HandleButton(Event_t *event);/// Handle mouse
button event in vertical splitter.
   virtual Bool_t HandleMotion(Event_t *event);/// Handle mouse
motion event in vertical splitter.
   virtual void   SavePrimitive(std::ostream &out, Option_t *opt
ion = "");
/// Save a splitter widget as a C++ statement(s) on output strea
m out.

   void LayoutHeader(TGFrame *f);  //*SIGNAL*  /// Emit LayoutFe
ader() signal.
   void LayoutListView();  //*SIGNAL*  /// Emit LayoutListView()
 signal.
   void ButtonPressed();   //*SIGNAL*  /// Emit ButtonPressed()
signal.
   void ButtonReleased();  //*SIGNAL*  /// Emit ButtonReleased()
 signal.
   void DoubleClicked(TGVFileSplitter* frame);  //*SIGNAL*  ///
Emit DoubleClicked() signal.
```

code

example

TGStatusBar

Provides a StatusBar widget.

TGStatusBar 继承 TGHorizontalFrame ,friend TGStatusBarPart

class

```
   TGStatusBar(const TGWindow *p = 0, UInt_t w = 4, UInt_t h = 2
,
               UInt_t options = kSunkenFrame | kHorizontalFrame,
               Pixel_t back = GetDefaultFrameBackground());
/// Create a status bar widget. By default it consist of one par
t.
/// Multiple parts can be created using SetParts().

   virtual ~TGStatusBar();/// Delete status bar widget.

   virtual void DrawBorder();/// Draw the status bar border (inc
luding cute 3d corner).
   virtual void SetText(TGString *text, Int_t partidx = 0);
/// Set text in partition partidx in status bar. The TGString is
/// adopted by the status bar.

   virtual void SetText(const char *text, Int_t partidx = 0);///
 Set text in partion partidx in status bar.
           void AddText(const char *text, Int_t partidx = 0)
                  { SetText(text, partidx); }                 /
/*MENU*
   const char  *GetText(Int_t partidx = 0) const;/// return text
 in the part partidx
   virtual void SetParts(Int_t npart);                         /
/*MENU*
/// Divide the status bar in npart equal sized parts.

   virtual void SetParts(Int_t *parts, Int_t npart);
/// Divide the status bar in nparts. Size of each part is given
```

```
in parts
/// array (percentual).

   void         Draw3DCorner(Bool_t corner) { f3DCorner = corner
; }
   TGCompositeFrame *GetBarPart(Int_t npart) const;
/// Returns bar part. That allows to put in the bar part
/// something more interesting than text ;-)

   TGDimension GetDefaultSize() const;/// Return default size.

   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
/// Save a status bar widget as a C++ statement(s) on output str
eam out.
```

code

example

TGTab

The TGTab is user callable. The TGTabElement and TGTabLayout are is a service classes of the tab widget.

TGTabLayout 继承 TGLayoutManager

TGTab 继承 TGCompositeFrame, TGWidget

A tab widget contains a set of composite frames each with a little tab with a name (like a set of folders with tabs).

```
Clicking on a tab will bring the associated composite frame to t
he
front and generate the following event:

kC_COMMAND, kCM_TAB, tab id, 0.
```

TGTabElement 继承 TGFrame

class

TGTabLayout

```
   TGTabLayout(TGTab *main);

   virtual void Layout();
   virtual TGDimension GetDefaultSize() const;
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");
```

TGTab

```
   static FontStruct_t  GetDefaultFontStruct();
   static const TGGC   &GetDefaultGC();
```

```
   TGTab(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1,
         GContext_t norm = GetDefaultGC()(),
         FontStruct_t font = GetDefaultFontStruct(),
         UInt_t options = kChildFrame,
         Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGTab();

   virtual TGCompositeFrame *AddTab(TGString *text);
   virtual TGCompositeFrame *AddTab(const char *text);
   virtual void              AddTab(const char *text, TGComposit
eFrame *cf);
   virtual void              AddTab(TGString *text, TGCompositeF
rame *cf);

   virtual void              NewTab(const char *text = "tab");
// *MENU*icon=bld_newtab.png*
   virtual void              RemoveTab(Int_t tabIndex = -1,
                                       Bool_t storeRemoved = kTR
UE); // *MENU*icon=bld_removetab.png*
   virtual Bool_t            SetTab(Int_t tabIndex, Bool_t emit
= kTRUE);
   virtual Bool_t            SetTab(const char *name, Bool_t emi
t = kTRUE);
   virtual void              DrawBorder() { }

   TGCompositeFrame *GetContainer() const { return fContainer; }
   Int_t             GetCurrent() const { return fCurrent; }
   TGCompositeFrame *GetTabContainer(Int_t tabIndex) const;
   TGCompositeFrame *GetTabContainer(const char *name) const;
   TGTabElement     *GetTabTab(Int_t tabIndex) const;
   TGTabElement     *GetTabTab(const char *name) const;
   TGCompositeFrame *GetCurrentContainer() const { return GetTab
Container(fCurrent); }
   TGTabElement     *GetCurrentTab() const { return GetTabTab(fC
urrent); }
   UInt_t            GetTabHeight() const { return fTabh; }
   Int_t             GetNumberOfTabs() const;
   virtual void      SetEnabled(Int_t tabIndex, Bool_t on = kTRU
E);  //*MENU*
   virtual void      SetText(const char *text = "tab");
```

```
      //*MENU*icon=bld_rename.png*
   Bool_t            IsEnabled(Int_t tabIndex) const;

   virtual void      SavePrimitive(std::ostream &out, Option_t *
option = "");

   virtual void CloseTab(Int_t id) { Emit("CloseTab(Int_t)", id)
; }  //*SIGNAL*
   virtual void Removed(Int_t id) { Emit("Removed(Int_t)", id);
}    //*SIGNAL*
   virtual void Selected(Int_t id) { Emit("Selected(Int_t)", id)
; }  //*SIGNAL*
   virtual TGLayoutManager *GetLayoutManager() const;
```

TGTabElement

```
   TGTabElement(const TGWindow *p = 0, TGString *text = 0, UInt_
t w = 1, UInt_t h = 1,
                GContext_t norm = TGTab::GetDefaultGC()(),
                FontStruct_t font = TGTab::GetDefaultFontStruct(
),
                UInt_t options = kRaisedFrame,
                Pixel_t back = GetDefaultFrameBackground());
   virtual ~TGTabElement();

   virtual void        DrawBorder();
   virtual TGDimension GetDefaultSize() const;
   const TGString     *GetText() const { return fText; }
   const char         *GetString() const { return fText->GetStri
ng(); }
   virtual Bool_t      HandleButton(Event_t *event);
   void                SetText(TGString *text);
   virtual void        SetEnabled(Bool_t on = kTRUE) { fEnabled
= on; }
   Bool_t              IsEnabled() const { return fEnabled; }
   virtual void        SetEditDisabled(UInt_t) {}
   virtual void        ShowClose(Bool_t on = kTRUE);
   Bool_t              IsCloseShown() const { return fShowClose;
 }
   virtual void        SetActive(Bool_t on = kTRUE) { fActive =
on; }
   Bool_t              IsActive() const { return fActive; }
```

code

example

TGTable

TGTable 继承 TGCompositeFrame, TGWidget

A table used to visualize data from diffent sources.

class

```
   TGTable(const TGWindow *p = 0, Int_t id = 0,
           TVirtualTableInterface *interface = 0, UInt_t nrows =
50,
           UInt_t ncolumns = 20);
   virtual ~TGTable();

   virtual TObjArray *GetRow(UInt_t row);
   virtual TObjArray *GetColumn(UInt_t columns);

//    // Selection
//    virtual void Select(TGTableCell *celltl, TGTableCell *cell
br);
//    virtual void Select(UInt_t xcell1, UInt_t ycell1, UInt_t x
cell2, UInt_t ycell2);
//    virtual void SelectAll();
//    virtual void SelectRow(TGTableCell *cell);
//    virtual void SelectRow(UInt_t row);
//    virtual void SelectRows(UInt_t row, UInt_t nrows);
//    virtual void SelectColumn(TGTableCell *cell);
//    virtual void SelectColumn(UInt_t column);
//    virtual void SelectColumns(UInt_t column, UInt_t ncolumns);


//    virtual void SetSelectGC(TGGC *gc);
//    virtual void SetTextJustify(Int_t tmode);

   // Cells
   virtual const TGTableCell* GetCell(UInt_t i, UInt_t j) const;
   virtual TGTableCell* GetCell(UInt_t i, UInt_t j);
```

```
   virtual const TGTableCell* FindCell(TGString label) const;
   virtual TGTableCell* FindCell(TGString label);

   virtual void Show();

   // Because insertion and removal of columns in the middle of
a data
   // set is not yet supported in this design iteration, these m
ethods
   // have been commented out.

//    // Insert a range of columns or rows, if the label is empt
y, a
//    // default scheme will be used.
//    virtual void InsertRowBefore(UInt_t row, UInt_t nrows);
//    virtual void InsertRowBefore(TGString label, UInt_t nrows);

//    virtual void InsertRowAfter(UInt_t row, UInt_t nrows);
//    virtual void InsertRowAfter(TGString label, UInt_t nrows);
//    virtual void InsertRowAt(UInt_t row, UInt_t nrows = 1);
//    virtual void InsertRowAt(TGString label, UInt_t nrows);

//    virtual void InsertColumnBefore(UInt_t column, UInt_t ncol
umns);
//    virtual void InsertColumnBefore(TGString label, UInt_t nco
lumns);
//    virtual void InsertColumnAfter(UInt_t column, UInt_t ncolu
mns);
//    virtual void InsertColumnAfter(TGString label, UInt_t ncol
umns);
//    virtual void InsertColumnAt(UInt_t column, UInt_t ncolumns
 = 1);
//    virtual void InsertColumnAt(TGString label, UInt_t ncolumn
s);

//    // Remove rows or columns.
//    virtual void RemoveRows(UInt_t row, UInt_t nrows = 1);
//    virtual void RemoveColumns(UInt_t column, UInt_t ncolumns
= 1);
```

```
   // Update view
   virtual void UpdateView();

   // Getters
   virtual UInt_t       GetNTableRows() const;
   virtual UInt_t       GetNDataRows() const;
   virtual UInt_t       GetNTableColumns() const;
   virtual UInt_t       GetNDataColumns() const;
   virtual UInt_t       GetNTableCells() const;
   virtual UInt_t       GetNDataCells() const;
   virtual const  TTableRange *GetCurrentRange() const;

   virtual TVirtualTableInterface *GetInterface() { return fInte
rface; }

   virtual TGCanvas                 *GetCanvas() { return fCanva
s; }
   virtual const TGTableHeaderFrame *GetRHdrFrame() { return fRH
drFrame; }
   virtual const TGTableHeaderFrame *GetCHdrFrame() { return fCH
drFrame; }
   virtual const TGTableHeader      *GetRowHeader(const UInt_t r
ow) const;
   virtual TGTableHeader            *GetRowHeader(const UInt_t r
ow);
   virtual const TGTableHeader      *GetColumnHeader(const UInt_
t column) const;
   virtual TGTableHeader            *GetColumnHeader(const UInt_
t column);
   virtual TGTableHeader            *GetTableHeader();

//    virtual const TGGC*  GetSelectGC() const;
//    virtual const TGGC*  GetCellBckgndGC(TGTableCell *cell) co
nst;
//    virtual const TGGC*  GetCellBckgndGC(UInt_t row, UInt_t co
lumn) const;

   virtual Pixel_t GetRowBackground(UInt_t row) const;
   virtual Pixel_t GetHeaderBackground() const ;
```

```
    virtual void SetOddRowBackground(Pixel_t pixel);
    virtual void SetEvenRowBackground(Pixel_t pixel);
    virtual void SetHeaderBackground(Pixel_t pixel);
    virtual void SetDefaultColors();

    // Range manipulators
    virtual void MoveTable(Int_t rows, Int_t columns);
    virtual void GotoTableRange(Int_t xtl, Int_t ytl,
                                Int_t xbr, Int_t ybr);
    // Operators
    virtual TGTableCell* operator() (UInt_t row, UInt_t column);

    // Internal slots
    virtual void ScrollCHeaders(Int_t xpos);
    virtual void ScrollRHeaders(Int_t ypos);
    virtual void NextChunk();
    virtual void PreviousChunk();
    virtual void UserRangeChange();
    virtual void Goto();
    virtual void Update();
```

code

example

TGTableCell

TGTableContainer

TGTableHeader

TGTableLayout

TGTextEntry

```
// A TGTextEntry is a one line text input widget.
        //
//
        //
// Changing text in the text entry widget will generate the even
t:      //
// kC_TEXTENTRY, kTE_TEXTCHANGED, widget id, 0.
        //
// Hitting the enter key will generate:
        //
// kC_TEXTENTRY, kTE_ENTER, widget id, 0.
        //
// Hitting the tab key will generate:
        //
// kC_TEXTENTRY, kTE_TAB, widget id, 0.
        //
```

TGTextEntry 继承 TGFrame, TGWidget

class

```
   enum   EEchoMode { kNormal, kNoEcho, kPassword };
   enum   EInsertMode { kInsert, kReplace };

   static FontStruct_t  GetDefaultFontStruct();
   static const TGGC   &GetDefaultGC();

   TGTextEntry(const TGWindow *p, TGTextBuffer *text, Int_t id =
-1,
               GContext_t norm = GetDefaultGC()(),
               FontStruct_t font = GetDefaultFontStruct(),
               UInt_t option = kSunkenFrame | kDoubleBorder,
               Pixel_t back = GetWhitePixel());
```

```
   TGTextEntry(const TGWindow *parent = 0, const char *text = 0,
 Int_t id = -1);
   TGTextEntry(const TString &contents, const TGWindow *parent,
Int_t id = -1);

   virtual ~TGTextEntry();

   virtual  TGDimension GetDefaultSize() const;
   virtual  void        SetDefaultSize(UInt_t w, UInt_t h);

   virtual  void        AppendText(const char *text);
            void        Backspace();
            void        Clear(Option_t *option="");
            void        CursorLeft(Bool_t mark = kFALSE , Int_t
steps = 1);
            void        CursorRight(Bool_t mark = kFALSE , Int_t
 steps = 1);
            void        CursorWordForward(Bool_t mark = kFALSE);
            void        CursorWordBackward(Bool_t mark = kFALSE)
;
            void        Cut();
            void        Del();
            void        Deselect();
   virtual  void        DrawBorder();
            void        End(Bool_t mark = kFALSE);
   ETextJustification   GetAlignment() const       { return fAli
gnment; }
   TGTextBuffer        *GetBuffer() const { return fText; }
            Int_t       GetCursorPosition() const  { return fCur
sorIX; }
            TString     GetDisplayText() const;
   EEchoMode            GetEchoMode() const        { return fEch
oMode; }
   EInsertMode          GetInsertMode() const      { return fIns
ertMode; }
            TString     GetMarkedText() const;
            Int_t       GetMaxLength() const    { return fMaxLen
; }
   const    char       *GetText() const { return fText->GetStrin
```

```
g(); }
   virtual TGToolTip   *GetToolTip() const { return fTip; }
   virtual const char  *GetTitle() const { return GetText(); }
            Bool_t      HasMarkedText() const  { return fSelecti
onOn && (fStartIX != fEndIX); }
            Pixel_t     GetTextColor() const { return fNormGC.Ge
tForeground(); }
           FontStruct_t GetFontStruct() const { return fFontStru
ct; }
            void        Home(Bool_t mark = kFALSE);
   virtual  void        Insert(const char *);
   virtual  void        InsertText(const char *text, Int_t pos);
            Bool_t      IsFrameDrawn() const       { return fFra
meDrawn; }
            Bool_t      IsEdited() const           { return fEdi
ted; }
   virtual  void        Layout() { UpdateOffset(); }
            void        MarkWord(Int_t pos);
            Int_t       MaxMark() const { return fStartIX > fEnd
IX ? fStartIX : fEndIX; }
            Int_t       MinMark() const { return fStartIX < fEnd
IX ? fStartIX : fEndIX; }
            void        NewMark(Int_t pos);
            void        Remove();
   virtual  void        RemoveText(Int_t start, Int_t end);
   virtual  void        SetFont(TGFont *font, Bool_t local = kTR
UE);
   virtual  void        SetFont(FontStruct_t font, Bool_t local
= kTRUE);
   virtual  void        SetFont(const char *fontName, Bool_t loc
al = kTRUE);
   virtual  void        SetTextColor(Pixel_t color, Bool_t local
 = kTRUE);
   virtual  void        SetTextColor(TColor *color, Bool_t local
 = kTRUE);
   virtual  void        SetText(const char *text, Bool_t emit =
kTRUE);          //*MENU*
   virtual  void        SetToolTipText(const char *text, Long_t
delayms = 500);  //*MENU*
   virtual  void        SetMaxLength(Int_t maxlen);
```

```
                    //*MENU*
   virtual  void        SelectAll();
   virtual  void        SetAlignment(ETextJustification mode = k
TextLeft);        //*SUBMENU*
   virtual  void        SetInsertMode(EInsertMode mode = kInsert)
;                //*SUBMENU*
   virtual  void        SetEchoMode(EEchoMode mode = kNormal);
                    //*SUBMENU*
            void        SetEnabled(Bool_t flag = kTRUE) { SetSta
te( flag ); }    //*TOGGLE* *GETTER=IsEnabled
   virtual  void        SetCursorPosition(Int_t pos);
            void        SetEdited(Bool_t flag = kTRUE) { fEdited
 = flag; }
   virtual  void        SetFocus();
   virtual  void        SetFrameDrawn(Bool_t flag = kTRUE);
   virtual  void        SetState(Bool_t state);
   virtual  void        SetTitle(const char *label) { SetText(la
bel); }
   virtual  void        SetForegroundColor(Pixel_t fore) { SetTe
xtColor(fore, kFALSE); }
   Pixel_t              GetForeground() const { return fNormGC.G
etForeground(); }
   Bool_t               HasOwnFont() const { return fHasOwnFont;
 }

   virtual  void        SavePrimitive(std::ostream &out, Option_
t *option = "");

   virtual  Bool_t      HandleButton(Event_t *event);
   virtual  Bool_t      HandleDoubleClick(Event_t *event);
   virtual  Bool_t      HandleCrossing(Event_t *event);
   virtual  Bool_t      HandleMotion(Event_t *event);
   virtual  Bool_t      HandleKey(Event_t *event);
   virtual  Bool_t      HandleFocusChange(Event_t *event);
   virtual  Bool_t      HandleSelection(Event_t *event);
   virtual  Bool_t      HandleSelectionClear(Event_t *event);
   virtual  Bool_t      HandleSelectionRequest(Event_t *event);
   virtual  Bool_t      HandleTimer(TTimer *t);
   virtual  Bool_t      HandleConfigureNotify(Event_t *event);
```

```
   virtual  void        TextChanged(const char *text = 0);
//*SIGNAL*
   virtual  void        ReturnPressed();
//*SIGNAL*
   virtual  void        TabPressed();
//*SIGNAL*
   virtual  void        ShiftTabPressed();
//*SIGNAL*
   virtual  void        CursorOutLeft();
//*SIGNAL*
   virtual  void        CursorOutRight();
//*SIGNAL*
   virtual  void        CursorOutUp();
//*SIGNAL*
   virtual  void        CursorOutDown();
//*SIGNAL*
   virtual  void        DoubleClicked();
//*SIGNAL*
```

code

```
// TGTextEntry
// 单个文本框
TGTextEntry *fTestText = new TGTextEntry(frame, new TGTextBuffer(
100));
fTestText->SetToolTipText("This is a text entry widget");
fTestText->Resize(300, fTestText->GetDefaultHeight());
frame->AddFrame(fTestText, new TGLayoutHints(kLHintsTop | kLHint
sLeft, 10, 2, 2, 2));
// fTestText->Clear();//清除内容
std::string command = fTestText->GetText();
```

example

TGTextViewStream

TGTextViewStreamBuf 继承 std::streambuf

TGTextViewostream 继承 TGTextView, std::ostream

A TGTextViewStream is a text viewer widget. It is a specialization of TGTextView and std::ostream, and it uses a TGTextViewStreamBuf, who inherits from std::streambuf, allowing to stream text directly to the text view in a cout-like fashion

class

TGTextViewStreamBuf

```
   TGTextViewStreamBuf(TGTextView *textview);
   virtual ~TGTextViewStreamBuf() { }
```

TGTextViewostream

```
   TGTextViewostream(const TGWindow* parent = 0, UInt_t w = 1, U
Int_t h = 1,
                     Int_t id = -1, UInt_t sboptions = 0,
                     Pixel_t back = TGTextView::GetWhitePixel())
;
   TGTextViewostream(const TGWindow *parent, UInt_t w, UInt_t h,
                     TGText *text, Int_t id, UInt_t sboptions, U
Long_t back);
   TGTextViewostream(const TGWindow *parent, UInt_t w, UInt_t h,
                     const char *string, Int_t id, UInt_t sbopti
ons,
                     ULong_t back);
   virtual ~TGTextViewostream() { }
```

code

```
#include "TGTextViewStream.h"

// TGTextViewStream
// 数据流输出框
TGTextViewostream *fTextView;
TGTextEntry       *fCommand;
fTextView = new TGTextViewostream(frame1, 500, 300);
frame1->AddFrame(fTextView, new TGLayoutHints(kLHintsExpandX | k
LHintsExpandY, 5, 5, 5, 0));
fCommand = new TGTextEntry(frame2, (const char *)"", 20);
fCommand->Connect("ReturnPressed()", "TextViewMainFrame", this,
"HandleReturn()");
frame2->AddFrame(fCommand, new TGLayoutHints(kLHintsExpandX, 5, 5
, 5, 5));
void HandleReturn()
{
   std::string line;
   std::string command = fCommand->GetText();
   *fTextView << gSystem->GetFromPipe(command.c_str()).Data() <<
std::endl;
   fTextView->ShowBottom();
   fCommand->Clear();
}
```

example

TGTripleSlider

TripleSlider inherit from DoubleSlider widgets and allow easy selection of a min, max and pointer value out of a range.
The pointer position can be constrained to edges of slider and / or can be relative to the slider position.

To change the min value press the mouse near to the left / bottom edge of the slider.
To change the max value press the mouse near to the right / top edge of the slider.
To change both values simultaneously press the mouse near to the center of the slider.
To change pointer value press the mouse on the pointer and drag it to the desired position

```
Dragging the slider will generate the event:

kC_VSLIDER, kSL_POS, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_POS, slider id, 0  (for horizontal slider)


Pressing the mouse will generate the event:

kC_VSLIDER, kSL_PRESS, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_PRESS, slider id, 0  (for horizontal slider)


Releasing the mouse will generate the event:

kC_VSLIDER, kSL_RELEASE, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_RELEASE, slider id, 0  (for horizontal slider)


Moving the pointer will generate the event:

kC_VSLIDER, kSL_POINTER, slider id, 0  (for vertical slider)

kC_HSLIDER, kSL_POINTER, slider id, 0  (for horizontal slider)
```

Use the functions GetMinPosition(), GetMaxPosition() and GetPosition() to retrieve the position of the slider.
Use the function GetPointerPosition() to retrieve the position of the pointer

TGTripleVSlider 继承 TGDoubleVSlider

TGTripleHSlider 继承 TGDoubleHSlider

class

TGTripleVSlider

```
   TGTripleVSlider(const TGWindow *p = 0, UInt_t h = 1, UInt_t t
ype = 1, Int_t id = -1,
                    UInt_t options = kVerticalFrame,
                    Pixel_t back = GetDefaultFrameBackground(),
                    Bool_t reversed = kFALSE,
                    Bool_t mark_ends = kFALSE,
                    Bool_t constrained = kTRUE,
                    Bool_t relative = kFALSE);

   virtual ~TGTripleVSlider();

   virtual void      PointerPositionChanged() { Emit("PointerPos
itionChanged()"); } //*SIGNAL*
   virtual void      DrawPointer();
   virtual Float_t   GetPointerPosition() const {
      if (fReversedScale) return fVmin + fVmax - fSCz;
      else return fSCz;
   }
   virtual Bool_t    HandleButton(Event_t *event);
   virtual Bool_t    HandleConfigureNotify(Event_t* event);
   virtual Bool_t    HandleMotion(Event_t *event);
   virtual void      SetConstrained(Bool_t on = kTRUE);
   virtual void      SetPointerPosition(Float_t pos);
   virtual void      SetRelative(Bool_t rel = kTRUE) { fRelative
 = rel; }
   virtual void      SavePrimitive(std::ostream &out, Option_t *
option = "");
```

TGTripleHSlider

```
   TGTripleHSlider(const TGWindow *p = 0, UInt_t w = 1, UInt_t t
ype = 1, Int_t id = -1,
                   UInt_t options = kHorizontalFrame,
                   Pixel_t back = GetDefaultFrameBackground(),
                   Bool_t reversed = kFALSE,
                   Bool_t mark_ends = kFALSE,
                   Bool_t constrained = kTRUE,
                   Bool_t relative = kFALSE);

   virtual ~TGTripleHSlider();

   virtual void      PointerPositionChanged() { Emit("PointerPos
itionChanged()"); } //*SIGNAL*
   virtual void      DrawPointer();
   virtual Float_t   GetPointerPosition() const {
      if (fReversedScale) return fVmin + fVmax - fSCz;
      else return fSCz;
   }
   virtual Bool_t    HandleButton(Event_t *event);
   virtual Bool_t    HandleConfigureNotify(Event_t* event);
   virtual Bool_t    HandleMotion(Event_t *event);
   virtual void      SetConstrained(Bool_t on = kTRUE);
   virtual void      SetPointerPosition(Float_t pos);
   virtual void      SetRelative(Bool_t rel = kTRUE) { fRelative
 = rel; }
   virtual void      SavePrimitive(std::ostream &out, Option_t *
option = "");
```

code

```
#include "TGTripleSlider.h"

// TGTripleSlider
// 可调节范围，左右边界外加一个中间值
TGTripleHSlider *fHslider1;
fHslider1 = new TGTripleHSlider(frame, 190, kDoubleScaleBoth, ID
s,
                kHorizontalFrame,
                GetDefaultFrameBackground(),
                kFALSE, kFALSE, kFALSE, kFALSE);
fHslider1->Connect("PointerPositionChanged()", "TTripleSliderDem
o", this, "DoSlider()");
fHslider1->Connect("PositionChanged()", "TTripleSliderDemo", this
, "DoSlider()");
fHslider1->SetRange(0.05,5.0);
frame->AddFrame(fHslider1, new TGLayoutHints(kLHintsTop | kLHint
sExpandX, 5, 5, 5, 5));
fHslider1->GetMinPosition();//Float_t
fHslider1->GetPointerPosition();//Float_t
fHslider1->GetMaxPosition());//Float_t
fHslider1->SetPointerPosition(atof("1.23"));
fHslider1->SetPosition(atof("5.0"), fHslider1->GetMaxPosition())
;
fHslider1->SetPosition(fHslider1->GetMinPosition(), atof("0.1");
fHslider1->SetConstrained(kTRUE);
fHslider1->SetRelative(kTRUE);
```

example

TGView

TGView 继承 TGCompositeFrame, TGWidget , friend TGViewFrame

A TGView provides the infrastructure for text viewer and editor widgets. It provides a canvas (TGViewFrame) and (optionally) a vertical and horizontal scrollbar and methods for marking and scrolling.

TGViewFrame 继承 TGCompositeFrame

class

TGView

```
   enum { kNoHSB = BIT(0), kNoVSB = BIT(1) };
   enum { kHorizontal = 0, kVertical = 1 };

   TGView(const TGWindow *p = 0, UInt_t w = 1, UInt_t h = 1, Int
_t id = -1,
          UInt_t xMargin = 0, UInt_t yMargin = 0,
          UInt_t options = kSunkenFrame | kDoubleBorder,
          UInt_t sboptions = 0,
          Pixel_t back = GetWhitePixel());

   virtual ~TGView();

   TGViewFrame   *GetCanvas() const { return fCanvas; }

   virtual void   Clear(Option_t * = "");
   virtual void   SetVisibleStart(Int_t newTop, Int_t direction)
;
   virtual void   ScrollCanvas(Int_t newTop, Int_t direction);
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_
t parm2);
   virtual void   DrawBorder();
   virtual void   Layout();
```

```
   virtual void   SetLayoutManager(TGLayoutManager*) { }
   virtual void   DrawRegion(Int_t x, Int_t y, UInt_t width, UIn
t_t height);

   virtual void ScrollToPosition(TGLongPosition newPos);
   void ScrollUp(Int_t pixels)
      { ScrollToPosition(TGLongPosition(fVisible.fX, fVisible.fY
 + pixels)); }
   void ScrollDown(Int_t pixels)
      { ScrollToPosition(TGLongPosition(fVisible.fX, fVisible.fY
 - pixels)); }
   void ScrollLeft(Int_t pixels)
      { ScrollToPosition(TGLongPosition(fVisible.fX + pixels, fV
isible.fY)); }
   void ScrollRight(Int_t  pixels)
      { ScrollToPosition(TGLongPosition(fVisible.fX - pixels, fV
isible.fY)); }

   virtual TGDimension GetDefaultSize() const { return TGDimensi
on(fWidth, fHeight); }
   TGDimension GetVirtualSize() const { return fVirtualSize; }
   TGLongPosition  GetScrollValue() const { return fScrollVal; }
   TGLongPosition  GetScrollPosition() const { return fVisible;
}

   TGLongPosition ToVirtual(TGLongPosition coord)  const { return
 coord + fVisible; }
   TGLongPosition ToPhysical(TGLongPosition coord) const { return
 coord - fVisible; }

   virtual Bool_t HandleButton(Event_t *event);
   virtual Bool_t HandleExpose(Event_t *event);

   virtual void   ChangeBackground(Pixel_t);
   virtual void   SetBackgroundColor(Pixel_t);
   virtual void   SetBackgroundPixmap(Pixmap_t p);
   virtual void   UpdateBackgroundStart();

   const TGGC &GetViewWhiteGC() { return fWhiteGC; }
```

TGViewFrame

```
   TGViewFrame(TGView *v, UInt_t w, UInt_t h, UInt_t options = 0
,
               Pixel_t back = GetWhitePixel());

   Bool_t HandleSelectionRequest(Event_t *event)
            { return fView->HandleSelectionRequest(event); }
   Bool_t HandleSelectionClear(Event_t *event)
            { return fView->HandleSelectionClear(event); }
   Bool_t HandleSelection(Event_t *event)
            { return fView->HandleSelection(event); }
   Bool_t HandleButton(Event_t *event)
            { return fView->HandleButton(event); }
   Bool_t HandleExpose(Event_t *event)
            { return fView->HandleExpose(event); }
   Bool_t HandleCrossing(Event_t *event)
            { return fView->HandleCrossing(event); }
   Bool_t HandleMotion(Event_t *event)
            { return fView->HandleMotion(event); }
   Bool_t HandleKey(Event_t *event)
            { return fView->HandleKey(event); }
   Bool_t HandleDoubleClick(Event_t *event)
            { return fView->HandleDoubleClick(event); }
```

code

example

TGWidget

The widget base class. It is light weight (all inline service methods) and is typically used as mixin class (via multiple inheritance), see for example TGButton.

class

```
//--- Text justification modes

enum ETextJustification {
   kTextLeft    = BIT(0),
   kTextRight   = BIT(1),
   kTextCenterX = BIT(2),
   kTextTop     = BIT(3),
   kTextBottom  = BIT(4),
   kTextCenterY = BIT(5)
};


//--- Widget status

enum EWidgetStatus {
   kWidgetWantFocus = BIT(0),
   kWidgetHasFocus  = BIT(1),
   kWidgetIsEnabled = BIT(2)
};
```

TGWidget

```
   TGWidget():
     fWidgetId(-1), fWidgetFlags(0), fMsgWindow(0), fCommand() {
 }
   TGWidget(Int_t id):
     fWidgetId(id), fWidgetFlags(0), fMsgWindow(0), fCommand() {
 }
   virtual ~TGWidget() { }

   Int_t         WidgetId() const { return fWidgetId; }
   Bool_t        IsEnabled() const { return (Bool_t)((fWidgetFla
gs & kWidgetIsEnabled) != 0); }
   Bool_t        HasFocus() const { return (Bool_t)((fWidgetFlag
s & kWidgetHasFocus) != 0); }
   Bool_t        WantFocus() const { return (Bool_t)((fWidgetFla
gs & kWidgetWantFocus) != 0); }
   virtual void  Associate(const TGWindow *w) { fMsgWindow = w;
}
   virtual void  SetCommand(const char *command) { fCommand = co
mmand; }
   const char   *GetCommand() const { return fCommand.Data(); }
```

code

example

TGWindow

TGWindow 继承 TGObject , friend TGClient
ROOT GUI Window base class.

TGUnknownWindowHandler 继承 TObject

Handle events for windows that are not part of the native ROOT GUI.
Typically windows created by Xt or Motif.

class

TGWindow

```
   enum  EEditMode {
      kEditEnable        = 0,          // allow edit of this win
dow
      kEditDisable       = BIT(0),     // disable edit of this w
indow
      kEditDisableEvents = BIT(1),     // window events cannot b
e editted
      kEditDisableGrab   = BIT(2),     // window grab cannot be
editted
      kEditDisableLayout = BIT(3),     // window layout cannot b
e editted
      kEditDisableResize = BIT(4),     // window size cannot be
editted
      kEditDisableHeight = BIT(5),     // window height cannot b
e editted
      kEditDisableWidth  = BIT(6),     // window width cannot be
 editted
      kEditDisableBtnEnable = BIT(7),  // window can handle mous
e button events
      kEditDisableKeyEnable = BIT(8)   // window can handle keyb
oard events
   };
```

```
   TGWindow(const TGWindow *p = 0, Int_t x = 0, Int_t y = 0,
            UInt_t w = 0, UInt_t h = 0, UInt_t border = 0,
            Int_t depth = 0,
            UInt_t clss = 0,
            void *visual = 0,
            SetWindowAttributes_t *attr = 0,
            UInt_t wtype = 0);
/// Create a new window. Parent p must exist otherwise the root
window
/// is taken as parent. No arguments specified results in values
 from
/// parent to be taken (or defaults).

   TGWindow(TGClient *c, Window_t id, const TGWindow *parent = 0
);/// Create a copy of a window.

   virtual ~TGWindow();/// Window destructor. Unregisters the wi
ndow.

   const TGWindow *GetParent() const { return fParent; }
   virtual const TGWindow *GetMainFrame() const;/// Returns top
level main frame.

   virtual void MapWindow() { gVirtualX->MapWindow(fId); }
   virtual void MapSubwindows() { gVirtualX->MapSubwindows(fId);
 }
   virtual void MapRaised() { gVirtualX->MapRaised(fId); }
   virtual void UnmapWindow() { gVirtualX->UnmapWindow(fId); }
   virtual void DestroyWindow() { gVirtualX->DestroyWindow(fId);
 }
   virtual void DestroySubwindows() { gVirtualX->DestroySubwindo
ws(fId); }
   virtual void RaiseWindow() { gVirtualX->RaiseWindow(fId); }
   virtual void LowerWindow() { gVirtualX->LowerWindow(fId); }
   virtual void IconifyWindow() { gVirtualX->IconifyWindow(fId);
 }
   virtual void ReparentWindow(const TGWindow *p, Int_t x = 0, I
nt_t y = 0);
/// Reparent window, make p the new parent and position the wind
ow at
```

```
/// position (x,y) in new parent.

   virtual void RequestFocus() { gVirtualX->SetInputFocus(fId);
}

   virtual void SetBackgroundColor(Pixel_t color)
                   { gVirtualX->SetWindowBackground(fId, color);
}
   virtual void SetBackgroundPixmap(Pixmap_t pixmap)
                   { gVirtualX->SetWindowBackgroundPixmap(fId, pi
xmap); }

   virtual Bool_t HandleExpose(Event_t *event)
                   { if (event->fCount == 0) fClient->NeedRedraw(
this); return kTRUE; }
   virtual Bool_t HandleEvent(Event_t *) { return kFALSE; }
   virtual Bool_t HandleTimer(TTimer *) { return kFALSE; }
   virtual Bool_t HandleIdleEvent(TGIdleHandler *) { return kFAL
SE; }

   virtual void   Move(Int_t x, Int_t y);/// Move the window.
   virtual void   Resize(UInt_t w, UInt_t h);/// Resize the wind
ow.
   virtual void   MoveResize(Int_t x, Int_t y, UInt_t w, UInt_t
h);/// Move and resize the window.
   virtual Bool_t IsMapped();/// Returns kTRUE if window is mapp
ed on screen, kFALSE otherwise.
   virtual Bool_t IsEditable() const { return (fClient->GetRoot(
) == this); }
   virtual UInt_t GetEditDisabled() const { return fEditDisabled
; }
   virtual void   SetEditDisabled(UInt_t on = kEditDisable) { fE
ditDisabled = on; }
   virtual void   SetEditable(Bool_t on = kTRUE)
                   { if (!(fEditDisabled & kEditDisable)) fClient
->SetRoot(on ? this : 0); }
   virtual Int_t  MustCleanup() const { return 0; }
   virtual void   Print(Option_t *option="") const;
/// Print window id.
/// If option is "tree" - print all parent windows tree
```

```
   virtual void        SetWindowName(const char *name = 0);/// S
et window name.
   virtual const char *GetName() const;/// Return unique name, u
sed in SavePrimitive methods.
   virtual void        SetName(const char *name) { fName = name;
 }

   virtual void   SetMapSubwindows(Bool_t /*on*/) {  }
   virtual Bool_t IsMapSubwindows() const { return kTRUE; }

   static Int_t        GetCounter();
/// Return global window counter (total number of created window
s).
```

```
   TGUnknownWindowHandler() { }
   virtual ~TGUnknownWindowHandler() { }

   virtual Bool_t HandleEvent(Event_t *) = 0;
```

code

example

TQObject

This is the ROOT implementation of the Qt object communication mechanism (see also http:www.troll.no/qt/metaobjects.html)

Signals and slots are used for communication between objects. When an object has changed in some way that might be interesting for the outside world, it emits a signal to tell whoever is listening. All slots that are connected to this signal will be activated (called). It is even possible to connect a signal directly to another signal (this will emit the second signal immediately whenever the first is emitted.) There is no limitation on the number of slots that can be connected to a signal.
The slots will be activated in the order they were connected to the signal. This mechanism allows objects to be easily reused, because the object that emits a signal does not need to know to what the signals are connected to.
Together, signals and slots make up a powerfull component programming mechanism.

TQObject friend TQConnection

TQObjSender 继承 TQObject

class

TQObject

```
   TQObject();
/// TQObject Constructor.
/// Comment:
///  - In order to minimize memory allocation fListOfSignals and
///    fListOfConnections are allocated only if it is neccesary
///  - When fListOfSignals/fListOfConnections are empty they will

///    be deleted
```

```
   virtual ~TQObject();
/// TQObject Destructor.
///    - delete all connections and signal list

   TList   *GetListOfClassSignals() const;/// Returns pointer to
 list of signals of this class.
   TList   *GetListOfSignals() const { return fListOfSignals; }
   TList   *GetListOfConnections() const { return fListOfConnect
ions; }

   Bool_t   AreSignalsBlocked() const { return fSignalsBlocked;
}
   Bool_t   BlockSignals(Bool_t b)
            { Bool_t ret = fSignalsBlocked; fSignalsBlocked = b;
return ret; }

   void  CollectClassSignalLists(TList& list, TClass* cls);
/// Collect class signal lists from class cls and all its
/// base-classes.
/// The recursive traversal is not performed for classes not
/// deriving from TQClass.

   template <typename... T> void EmitVA(const char *signal_name,
 Int_t /* nargs */, const T&... params);
   // void  EmitVA(const char *signal, Int_t nargs, ...);
   void  EmitVA(const char *signal, Int_t nargs, va_list va) = d
elete;
   void  Emit(const char *signal);
/// Acitvate signal without args.

   void  Emit(const char *signal, Long_t *paramArr);
/// Emit a signal with a varying number of arguments,
/// paramArr is an array of the parameters.
/// Note: any parameter should be converted to long type.

   void  Emit(const char *signal, const char *params);
/// Activate signal with parameter text string.

   void  Emit(const char *signal, Double_t param);
```

```
/// Activate signal with single parameter.

   void  Emit(const char *signal, Long_t param);
/// Activate signal with single parameter.

   void  Emit(const char *signal, Long64_t param);
/// Activate signal with single parameter.

   void  Emit(const char *signal, Bool_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, Char_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, UChar_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, Short_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, UShort_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, Int_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, UInt_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, ULong_t param)
         { Emit(signal, (Long_t)param); }
   void  Emit(const char *signal, ULong64_t param)
         { Emit(signal, (Long64_t) param); }
   void  Emit(const char *signal, Float_t param)
         { Emit(signal, (Double_t)param); }

   Bool_t Connect(const char *signal,
                  const char *receiver_class,
                  void *receiver,
                  const char *slot);
/// Non-static method is used to connect from the signal
/// of this object to the receiver slot.
/// Warning! No check on consistency of sender/receiver
/// classes/methods.
/// This method makes possible to have connection/signals from
/// interpreted class. See also RQ_OBJECT.h.
```

```
   Bool_t Disconnect(const char *signal = 0,
                     void *receiver = 0,
                     const char *slot = 0);
/// Disconnects signal of this object from slot of receiver.
/// Equivalent to Disconnect(this, signal, receiver, slot)

   virtual void   HighPriority(const char *signal_name,
                               const char *slot_name = 0);
/// 1. If slot_name = 0 => makes signal defined by the signal_na
me
///    to be the first in the fListOfSignals, this decreases
///    the time for lookup.
/// 2. If slot_name != 0 => makes slot defined by the slot_name
///    to be executed first when signal_name is emitted.
/// Signal name is not compressed.

   virtual void   LowPriority(const char *signal_name,
                              const char *slot_name = 0);
/// 1. If slot_name = 0 => makes signal defined by the signal_na
me
///    to be the last in the fListOfSignals, this increase the t
ime
///    for lookup.
/// 2. If slot_name != 0 => makes slot defined by the slot_name
///    to  be executed last when signal_name is emitted.
/// Signal name is not compressed.

   virtual Bool_t HasConnection(const char *signal_name) const;
/// Return true if there is any object connected to this signal.
/// Only checks for object signals.

   virtual Int_t  NumberOfSignals() const;
/// Return number of signals for this object.
/// Only checks for object signals.

   virtual Int_t  NumberOfConnections() const;/// Return number
of connections for this object.
   virtual void   Connected(const char * /*signal_name*/) { }
   virtual void   Disconnected(const char * /*signal_name*/) { }
```

```
   virtual void   Destroyed()
                  { Emit("Destroyed()"); }                // *S
IGNAL*
   virtual void   ChangedBy(const char *method)
                  { Emit("ChangedBy(char*)", method); }    // *S
IGNAL*
   virtual void   Message(const char *msg)
                  { Emit("Message(char*)", msg); }         // *S
IGNAL*

   static Bool_t  Connect(TQObject *sender,
                          const char *signal,
                          const char *receiver_class,
                          void *receiver,
                          const char *slot);
/// Create connection between sender and receiver.
/// Signal and slot string must have a form:
///    "Draw(char*, Option_t* ,Int_t)"
/// All blanks and "const" words will be removed,
/// Warning:
///  If receiver is class not derived from TQObject and going to
 be
///  deleted, disconnect all connections to this receiver.
///  In case of class derived from TQObject it is done automatic
ally.


   static Bool_t  Connect(const char *sender_class,
                          const char *signal,
                          const char *receiver_class,
                          void *receiver,
                          const char *slot);
/// This method allows to make a connection from any object
/// of the same class to a single slot.
/// Signal and slot string must have a form:
///    "Draw(char*, Option_t* ,Int_t)"
/// All blanks and "const" words will be removed,
/// Warning:
///  If receiver class not derived from TQObject and going to be
///  deleted, disconnect all connections to this receiver.
```

```
///  In case of class derived from TQObject it is done automatic
ally.

   static Bool_t  Disconnect(TQObject *sender,
                             const char *signal = 0,
                             void *receiver = 0,
                             const char *slot = 0);
/// Disconnects signal in object sender from slot_method in
/// object receiver. For objects derived from TQObject signal-sl
ot
/// connection is removed when either of the objects involved
/// are destroyed.
/// Disconnect() is typically used in three ways, as the followi
ng
/// examples shows:
///  - Disconnect everything connected to an object's signals:
///       Disconnect(myObject);
///  - Disconnect everything connected to a signal:
///       Disconnect(myObject, "mySignal()");
///  - Disconnect a specific receiver:
///       Disconnect(myObject, 0, myReceiver, 0);
/// 0 may be used as a wildcard in three of the four arguments,
/// meaning "any signal", "any receiving object" or
/// "any slot in the receiving object", respectively.
/// The sender has no default and may never be 0
/// (you cannot disconnect signals from more than one object).
/// If signal is 0, it disconnects receiver and slot_method
/// from any signal. If not, only the specified signal is
/// disconnected.
/// If  receiver is 0, it disconnects anything connected to sign
al.
/// If not, slots in objects other than receiver are not
/// disconnected
/// If slot_method is 0, it disconnects anything that is connect
ed
/// to receiver.  If not, only slots named slot_method will be
/// disconnected, and all other slots are left alone.
/// The slot_method must be 0 if receiver is left out, so you
/// cannot disconnect a specifically-named slot on all objects.
```

```
   static Bool_t  Disconnect(const char *class_name,
                             const char *signal,
                             void *receiver = 0,
                             const char *slot = 0);
/// Disconnects "class signal". The class is defined by class_na
me.
/// See also Connect(class_name,signal,receiver,slot).

   static Bool_t  AreAllSignalsBlocked();/// Returns true if all
 signals are blocked.
   static Bool_t  BlockAllSignals(Bool_t b);
/// Block or unblock all signals. Returns the previous block sta
tus.
```

TQObjSender

```
   TQObjSender() : TQObject(), fSender(0), fSenderClass() { }
   virtual ~TQObjSender() { Disconnect(); }

   virtual void SetSender(void *sender) { fSender = sender; }
   void SetSenderClassName(const char *sclass = "") { fSenderCla
ss = sclass; }
```

code

```
// void TQObject::Emit(const char *signal_name)

theButton->Emit("Clicked()");
```

```
// void TQObject::Emit(const char *signal_name, Long_t param)

theButton->Emit("Clicked(int)",id)
```

```
// void TQObject::Emit(const char *signal_name, Long64_t param)

theButton->Emit("Progress(Long64_t)",processed)
```

```
// void TQObject::Emit(const char *signal_name, Double_t param)

theButton->Emit("Scale(float)",factor)
```

```
// void TQObject::Emit(const char *signal_name, const char *para
ms)

myObject->Emit("Error(char*)","Fatal error");
```

```
// void TQObject::Emit(const char *signal_name, Long_t *paramArr)


TQObject *processor; // data processor
TH1F     *hist;      // filled with processor results

processor->Connect("Evaluated(Float_t,Float_t)",
                   "TH1F",hist,"Fill12(Axis_t,Axis_t)");

Long_t args[2];
args[0] = (Long_t)processor->GetValue(1);
args[1] = (Long_t)processor->GetValue(2);

processor->Emit("Evaluated(Float_t,Float_t)",args);
```

```
// Bool_t TQObject::Connect(TQObject *sender,
//                          const char *signal,
//                          const char *cl,
//                          void *receiver,
//                          const char *slot)

/// cl != 0 - class name, it can be class with or
///           without dictionary, e.g interpreted class.
TGButton *myButton;
TH2F     *myHist;
TQObject::Connect(myButton,"Clicked()",
                  "TH2F", myHist,"Draw(Option_t*)");

/// cl == 0 - corresponds to function (interpereted or global)
///           the name of the function is defined by the slot st
ring,
///           parameter receiver should be 0.
TGButton *myButton;
TH2F     *myHist;
TQObject::Connect(myButton,"Clicked()",
                  0, 0,"hsimple()");
```

```
// Bool_t TQObject::Connect(const char *class_name,
//                          const char *signal,
//                          const char *cl,
//                          void *receiver,
//                          const char *slot)

/// cl != 0 - class name, it can be class with or
///           without dictionary, e.g interpreted class.

TGButton *myButton;
TH2F     *myHist;

TQObject::Connect("TGButton", "Clicked()",
                  "TH2F", myHist, "Draw(Option_t*)");

/// cl == 0 - corresponds to function (interpereted or global)
///           the name of the function is defined by the slot st
ring,
///           parameter receiver should be 0.

TGButton *myButton;
TH2F     *myHist;
TQObject::Connect("TGButton", "Clicked()",
                  0, 0, "hsimple()");
```

example

TRootEmbeddedCanvas

TRootEmbeddedCanvas 继承 TGCanvas , friend TRootEmbeddedContainer

This class creates a TGCanvas in which a TCanvas is created. Use GetCanvas() to get a pointer to the TCanvas.

class

```
   TRootEmbeddedCanvas(const char *name = 0, const TGWindow *p =
0, UInt_t w = 10,
            UInt_t h = 10, UInt_t options = kSunkenFrame | kDoub
leBorder,
            Pixel_t back = GetDefaultFrameBackground());
   virtual ~TRootEmbeddedCanvas();

   void       AdoptCanvas(TCanvas *c);
   TCanvas   *GetCanvas() const { return fCanvas; }
   Int_t      GetCanvasWindowId() const { return fCWinId; }
   Bool_t     GetAutoFit() const { return fAutoFit; }
   void       SetAutoFit(Bool_t fit = kTRUE) { fAutoFit = fit; }
   virtual void SavePrimitive(std::ostream &out, Option_t *optio
n = "");

   virtual Bool_t HandleDNDDrop(TDNDData *data);
   virtual Atom_t HandleDNDPosition(Int_t /*x*/, Int_t /*y*/, At
om_t action,
                                    Int_t /*xroot*/, Int_t /*yro
ot*/);
   virtual Atom_t HandleDNDEnter(Atom_t * typelist);
   virtual Bool_t HandleDNDLeave();
```

code

```
#include "TRootEmbeddedCanvas.h"

// TRootEmbeddedCanvas
// 画板
TRootEmbeddedCanvas *fCanvas= new TRootEmbeddedCanvas("Canvas",
frame, 600, 400);
TGLayoutHints *fLcan = new TGLayoutHints(kLHintsExpandX | kLHint
sExpandY, 10, 10, 10, 10);
frame->AddFrame(fCanvas, fLcan);
fCanvas->GetCanvas()->SetFillColor(33);
fCanvas->GetCanvas()->SetFrameFillColor(41);
fCanvas->GetCanvas()->SetBorderMode(0);
fCanvas->GetCanvas()->SetGrid();
fCanvas->GetCanvas()->SetLogy();
fCanvas->GetCanvas()->Modified();
fCanvas->GetCanvas()->Update();
```

example

TDirectory

继承TNamed

Describe directory structure in memory.

class

```
   static  void        AddDirectory(Bool_t add=kTRUE);
   static  Bool_t      AddDirectoryStatus();
   virtual void        Append(TObject *obj, Bool_t replace = kFA
LSE);
   virtual void        Add(TObject *obj, Bool_t replace = kFALSE)
 { Append(obj,replace); }
   virtual Int_t       AppendKey(TKey *) {return 0;}
   virtual void        Browse(TBrowser *b);
   virtual void        Build(TFile* motherFile = 0, TDirectory*
motherDir = 0);
   virtual void        Clear(Option_t *option="");
   virtual TObject    *CloneObject(const TObject *obj, Bool_t au
toadd = kTRUE);
   virtual void        Close(Option_t *option="");
   static TDirectory *&CurrentDirectory();  // Return the curren
t directory for this thread.
   virtual void        Copy(TObject &) const { MayNotUse("Copy(T
Object &)"); }
   virtual Bool_t      cd(const char *path = 0);
   virtual void        DeleteAll(Option_t *option="");
   virtual void        Delete(const char *namecycle="");
   virtual void        Draw(Option_t *option="");
   virtual TKey       *FindKey(const char * /*keyname*/) const {
return 0;}
   virtual TKey       *FindKeyAny(const char * /*keyname*/) const
 {return 0;}
   virtual TObject    *FindObject(const char *name) const;
   virtual TObject    *FindObject(const TObject *obj) const;
   virtual TObject    *FindObjectAny(const char *name) const;
```

```
   virtual TObject    *FindObjectAnyFile(const char * /*name*/)
const {return 0;}
   virtual TObject    *Get(const char *namecycle);
   virtual TDirectory *GetDirectory(const char *namecycle, Bool_
t printError = false, const char *funcname = "GetDirectory");
   template <class T> inline void GetObject(const char* namecycl
e, T*& ptr) // See TDirectory::Get for information
      {
         ptr = (T*)GetObjectChecked(namecycle,TBuffer::GetClass(
typeid(T)));
      }
   virtual void       *GetObjectChecked(const char *namecycle, c
onst char* classname);
   virtual void       *GetObjectChecked(const char *namecycle, c
onst TClass* cl);
   virtual void       *GetObjectUnchecked(const char *namecycle)
;
   virtual Int_t       GetBufferSize() const {return 0;}
   virtual TFile      *GetFile() const { return 0; }
   virtual TKey       *GetKey(const char * /*name */, Short_t /*
 cycle */=9999) const {return 0;}
   virtual TList      *GetList() const { return fList; }
   virtual TList      *GetListOfKeys() const { return 0; }
   virtual TObject    *GetMother() const { return fMother; }
   virtual TDirectory *GetMotherDir() const { return fMother==0
? 0 : dynamic_cast<TDirectory*>(fMother); }
   virtual Int_t       GetNbytesKeys() const { return 0; }
   virtual Int_t       GetNkeys() const { return 0; }
   virtual Long64_t    GetSeekDir() const { return 0; }
   virtual Long64_t    GetSeekParent() const { return 0; }
   virtual Long64_t    GetSeekKeys() const { return 0; }
   virtual const char *GetPathStatic() const;
   virtual const char *GetPath() const;
   TUUID               GetUUID() const {return fUUID;}
   virtual Bool_t      IsFolder() const { return kTRUE; }
   virtual Bool_t      IsModified() const { return kFALSE; }
   virtual Bool_t      IsWritable() const { return kFALSE; }
   virtual void        ls(Option_t *option="") const;
   virtual TDirectory *mkdir(const char *name, const char *title=
"");
```

```
   virtual TFile      *OpenFile(const char * /*name*/, Option_t
* /*option*/ = "",
                            const char * /*ftitle*/ = "", Int_t
/*compress*/ = 1,
                            Int_t /*netopt*/ = 0) {return 0;}
   virtual void        Paint(Option_t *option="");
   virtual void        Print(Option_t *option="") const;
   virtual void        Purge(Short_t /*nkeep*/=1) {}
   virtual void        pwd() const;
   virtual void        ReadAll(Option_t * /*option*/="") {}
   virtual Int_t       ReadKeys(Bool_t /*forceRead*/=kTRUE) {ret
urn 0;}
   virtual Int_t       ReadTObject(TObject * /*obj*/, const char
 * /*keyname*/) {return 0;}
   virtual TObject    *Remove(TObject*);
   virtual void        RecursiveRemove(TObject *obj);
   virtual void        rmdir(const char *name);
   virtual void        Save() {}
   virtual Int_t       SaveObjectAs(const TObject * /*obj*/, con
st char * /*filename*/="", Option_t * /*option*/="") const;
   virtual void        SaveSelf(Bool_t /*force*/ = kFALSE) {}
   virtual void        SetBufferSize(Int_t /* bufsize */) {}
   virtual void        SetModified() {}
   virtual void        SetMother(TObject *mother) {fMother = (TO
bject*)mother;}
   virtual void        SetName(const char* newname);
   virtual void        SetTRefAction(TObject * /*ref*/, TObject
* /*parent*/) {}
   virtual void        SetSeekDir(Long64_t) {}
   virtual void        SetWritable(Bool_t) {}
   virtual Int_t       Sizeof() const {return 0;}
   virtual Int_t       Write(const char * /*name*/=0, Int_t /*op
t*/=0, Int_t /*bufsize*/=0){return 0;}
   virtual Int_t       Write(const char * /*name*/=0, Int_t /*op
t*/=0, Int_t /*bufsize*/=0) const {return 0;}
   virtual Int_t       WriteTObject(const TObject *obj, const ch
ar *name =0, Option_t * /*option*/="", Int_t /*bufsize*/ =0);
private:
           Int_t       WriteObject(void *obj, const char* name,
Option_t *option="", Int_t bufsize=0); // Intentionaly not imple
```

```
mented.
public:
   template <class T> inline Int_t WriteObject(const T* obj, con
st char* name, Option_t *option="", Int_t bufsize=0) // see TDir
ectory::WriteTObject or TDirectoryWriteObjectAny for explanation
      {
         return WriteObjectAny(obj,TBuffer::GetClass(typeid(T)),
name,option,bufsize);
      }
   virtual Int_t       WriteObjectAny(const void *, const char *
/*classname*/, const char * /*name*/, Option_t * /*option*/="",
Int_t /*bufsize*/ =0) {return 0;}
   virtual Int_t       WriteObjectAny(const void *, const TClass
 * /*cl*/, const char * /*name*/, Option_t * /*option*/="", Int_
t /*bufsize*/ =0) {return 0;}
   virtual void        WriteDirHeader() {}
   virtual void        WriteKeys() {}

   static Bool_t       Cd(const char *path);
   static void         DecodeNameCycle(const char *namecycle, ch
ar *name, Short_t &cycle, const size_t namesize = 0);
   static void         EncodeNameCycle(char *buffer, const char
*name, Short_t cycle);
```

TDirectoryFile

TDirectory

Describe directory structure in a ROOT file.

A ROOT file is structured in Directories (like a file system).
Each Directory has a list of Keys (see TKeys) and a list of objects in memory. A Key is a small object that describes the type and location of a persistent object in a file. The persistent object may be a directory.

class

```
   virtual void        Append(TObject *obj, Bool_t replace = kFA
LSE);
           void        Add(TObject *obj, Bool_t replace = kFALSE)
 { Append(obj,replace); }
           Int_t       AppendKey(TKey *key);
   virtual void        Browse(TBrowser *b);
           void        Build(TFile* motherFile = 0, TDirectory*
motherDir = 0);
   virtual TObject    *CloneObject(const TObject *obj, Bool_t au
toadd = kTRUE);
   virtual void        Close(Option_t *option="");
   virtual void        Copy(TObject &) const { MayNotUse("Copy(T
Object &)"); }
   virtual Bool_t      cd(const char *path = 0);
   virtual void        Delete(const char *namecycle="");
   virtual void        FillBuffer(char *&buffer);
   virtual TKey       *FindKey(const char *keyname) const;
   virtual TKey       *FindKeyAny(const char *keyname) const;
   virtual TObject    *FindObjectAny(const char *name) const;
   virtual TObject    *FindObjectAnyFile(const char *name) const
;
   virtual TObject    *Get(const char *namecycle);
   virtual TDirectory *GetDirectory(const char *apath, Bool_t pr
intError = false, const char *funcname = "GetDirectory");
```

```
   template <class T> inline void GetObject(const char* namecycl
e, T*& ptr) // See TDirectory::Get for information
      {
         ptr = (T*)GetObjectChecked(namecycle,TBuffer::GetClass(
typeid(T)));
      }
   virtual void       *GetObjectChecked(const char *namecycle, c
onst char* classname);
   virtual void       *GetObjectChecked(const char *namecycle, c
onst TClass* cl);
   virtual void       *GetObjectUnchecked(const char *namecycle)
;
   virtual Int_t       GetBufferSize() const;
   const TDatime      &GetCreationDate() const { return fDatimeC
; }
   virtual TFile      *GetFile() const { return fFile; }
   virtual TKey       *GetKey(const char *name, Short_t cycle=99
99) const;
   virtual TList      *GetListOfKeys() const { return fKeys; }
   const TDatime      &GetModificationDate() const { return fDat
imeM; }
   virtual Int_t       GetNbytesKeys() const { return fNbytesKey
s; }
   virtual Int_t       GetNkeys() const { return fKeys->GetSize(
); }
   virtual Long64_t    GetSeekDir() const { return fSeekDir; }
   virtual Long64_t    GetSeekParent() const { return fSeekParen
t; }
   virtual Long64_t    GetSeekKeys() const { return fSeekKeys; }
   Bool_t              IsModified() const { return fModified; }
   Bool_t              IsWritable() const { return fWritable; }
   virtual void        ls(Option_t *option="") const;
   virtual TDirectory *mkdir(const char *name, const char *title=
"");
   virtual TFile      *OpenFile(const char *name, Option_t *opti
on= "",
                              const char *ftitle = "", Int_t compr
ess = 1,
                              Int_t netopt = 0);
   virtual void        Purge(Short_t nkeep=1);
```

```
   virtual void        ReadAll(Option_t *option="");
   virtual Int_t       ReadKeys(Bool_t forceRead=kTRUE);
   virtual Int_t       ReadTObject(TObject *obj, const char *key
name);
   virtual void        ResetAfterMerge(TFileMergeInfo *);
   virtual void        rmdir(const char *name);
   virtual void        Save();
   virtual void        SaveSelf(Bool_t force = kFALSE);
   virtual Int_t       SaveObjectAs(const TObject *obj, const ch
ar *filename="", Option_t *option="") const;
   virtual void        SetBufferSize(Int_t bufsize);
   void                SetModified() {fModified = kTRUE;}
   void                SetSeekDir(Long64_t v) { fSeekDir = v; }
   virtual void        SetTRefAction(TObject *ref, TObject *pare
nt);
   void                SetWritable(Bool_t writable=kTRUE);
   virtual Int_t       Sizeof() const;
   virtual Int_t       Write(const char *name=0, Int_t opt=0, In
t_t bufsize=0);
   virtual Int_t       Write(const char *name=0, Int_t opt=0, In
t_t bufsize=0) const ;
   virtual Int_t       WriteTObject(const TObject *obj, const ch
ar *name=0, Option_t *option="", Int_t bufsize=0);
   virtual Int_t       WriteObjectAny(const void *obj, const char
 *classname, const char *name, Option_t *option="", Int_t bufsiz
e=0);
   virtual Int_t       WriteObjectAny(const void *obj, const TCl
ass *cl, const char *name, Option_t *option="", Int_t bufsize=0)
;
   virtual void        WriteDirHeader();
   virtual void        WriteKeys();
```

TObject

class

```
   virtual void        AppendPad(Option_t *option="");
   virtual void        Browse(TBrowser *b);
   virtual const char *ClassName() const;
   virtual void        Clear(Option_t * /*option*/ ="") { }
   virtual TObject    *Clone(const char *newname="") const;
   virtual Int_t       Compare(const TObject *obj) const;
   virtual void        Copy(TObject &object) const;
   virtual void        Delete(Option_t *option=""); // *MENU*
   virtual Int_t       DistancetoPrimitive(Int_t px, Int_t py);
   virtual void        Draw(Option_t *option="");
   virtual void        DrawClass() const; // *MENU*
   virtual TObject    *DrawClone(Option_t *option="") const; //
*MENU*
   virtual void        Dump() const; // *MENU*
   virtual void        Execute(const char *method,  const char *
params, Int_t *error=0);
   virtual void        Execute(TMethod *method, TObjArray *param
s, Int_t *error=0);
   virtual void        ExecuteEvent(Int_t event, Int_t px, Int_t
 py);
   virtual TObject    *FindObject(const char *name) const;
   virtual TObject    *FindObject(const TObject *obj) const;
   virtual Option_t   *GetDrawOption() const;
   virtual UInt_t      GetUniqueID() const;
   virtual const char *GetName() const;
   virtual const char *GetIconName() const;
   virtual Option_t   *GetOption() const { return ""; }
   virtual char       *GetObjectInfo(Int_t px, Int_t py) const;
   virtual const char *GetTitle() const;
   virtual Bool_t      HandleTimer(TTimer *timer);
   virtual ULong_t     Hash() const;
   virtual Bool_t      InheritsFrom(const char *classname) const
;
```

```
   virtual Bool_t      InheritsFrom(const TClass *cl) const;
   virtual void        Inspect() const; // *MENU*
   virtual Bool_t      IsFolder() const;
   virtual Bool_t      IsEqual(const TObject *obj) const;
   virtual Bool_t      IsSortable() const { return kFALSE; }
           Bool_t      IsOnHeap() const { return TestBit(kIsOnHe
ap); }
           Bool_t      IsZombie() const { return TestBit(kZombie
); }
   virtual Bool_t      Notify();
   virtual void        ls(Option_t *option="") const;
   virtual void        Paint(Option_t *option="");
   virtual void        Pop();
   virtual void        Print(Option_t *option="") const;
   virtual Int_t       Read(const char *name);
   virtual void        RecursiveRemove(TObject *obj);
   virtual void        SaveAs(const char *filename="",Option_t *
option="") const; // *MENU*
   virtual void        SavePrimitive(std::ostream &out, Option_t
 *option = "");
   virtual void        SetDrawOption(Option_t *option="");  // *
MENU*
   virtual void        SetUniqueID(UInt_t uid);
   virtual void        UseCurrentStyle();
   virtual Int_t       Write(const char *name=0, Int_t option=0,
 Int_t bufsize=0);
   virtual Int_t       Write(const char *name=0, Int_t option=0,
 Int_t bufsize=0) const;
```

TSystemDirectory

继承 TSystemFile

Describes an Operating System directory for the browser.

class

```
   TSystemDirectory();
   TSystemDirectory(const char *dirname, const char *path);

   virtual ~TSystemDirectory();

   virtual Bool_t IsFolder() const { return kTRUE; }
   virtual Bool_t IsDirectory(const char * = 0) const { return k
TRUE; }

   virtual void   Browse(TBrowser *b);
   virtual void   Edit() { }
   virtual TList *GetListOfFiles() const;
   virtual void   SetDirectory(const char *name);
   virtual void   Delete() {}
   virtual void   Copy(const char *) {}
   virtual void   Move(const char *) {}

   // dummy methods from TObject
   void        DrawClass() const { }
   TObject    *DrawClone(Option_t *) const { return 0; }
   void        SetDrawOption(Option_t *) { }
   void        SetName(const char *name) { TSystemFile::SetName(
name); }
   void        SetTitle(const char *title) { TSystemFile::SetTit
le(title); }
   void        Delete(Option_t *) { }
   void        Copy(TObject & ) const { }
```

TSystemFile

继承 TNamed

A TSystemFile describes an operating system file. The information is used by the browser (see TBrowser).

class

```
   TSystemFile();
   TSystemFile(const char *filename, const char *dirname);
   virtual ~TSystemFile();
   virtual void     Browse(TBrowser *b);
   virtual void     Rename(const char *name);      // *MENU*
   virtual void     Delete();                      // *MENU*
   virtual void     Copy(const char *to);          // *MENU*
   virtual void     Move(const char *to);          // *MENU*
   virtual void     Edit();                        // *MENU*

   virtual Bool_t   IsDirectory(const char *dir = 0) const;
   virtual void     SetIconName(const char *name) { fIconName =
name; }
   const char      *GetIconName() const { return fIconName.Data(
); }

   // dummy methods from TObject
   virtual void     Inspect() const;
   virtual void     Dump() const;

   void        DrawClass() const { }
   TObject    *DrawClone(Option_t *) const { return 0; }
   void        SetDrawOption(Option_t *) { }
   void        SetName(const char *name) { TNamed::SetName(name)
; }
   void        SetTitle(const char *title) { TNamed::SetTitle(ti
tle); }
   void        Delete(Option_t *) { }
   void        Copy(TObject & ) const { }
```

TTask

class

继承TNamed

Base class for recursive execution of tasks.

TTask is a base class that can be used to build a complex tree of Tasks. Each TTask derived class may contain other TTasks that can be executed recursively, such that a complex program can be dynamically built and executed by invoking the services of the top level Task or one of its subtasks.

Use the TTask::Add function to add a subtask to an existing TTask. To execute a TTask, one calls the ExecuteTask function. ExecuteTask will call recursively:

- the TTask::Exec function of the derived class
- TTask::ExecuteTasks to execute for each task the list of its subtasks.

If the top level task (see example below) is added to the list of Root browsable objects, the tree of tasks can be visualized by the Root browser. The browser can be used to start a task, set break points at the beginning of a task or when the task has completed. At a breakpoint, data structures generated by the execution up this point may be inspected asynchronously and then the execution can be resumed by selecting the "Continue" function of a task.

A Task may be active or inactive (controlled by TTask::SetActive). When a task is not active, its sub tasks are not executed.

A TTask tree may be made persistent, saving the status of all the tasks.

```
   TTask();
   TTask(const char* name, const char *title);
   virtual ~TTask();
   TTask(const TTask &task);
   TTask& operator=(const TTask& tt);

   virtual void  Abort();  // *MENU*
   virtual void  Add(TTask *task) {fTasks->Add(task);}
   virtual void  Browse(TBrowser *b);
   virtual void  CleanTasks();
   virtual void  Clear(Option_t *option="");
   virtual void  Continue(); // *MENU*
   virtual void  Exec(Option_t *option);
   virtual void  ExecuteTask(Option_t *option="0");  // *MENU*
   virtual void  ExecuteTasks(Option_t *option);
   Int_t         GetBreakin() const { return fBreakin; }
   Int_t         GetBreakout() const { return fBreakout; }
         Bool_t  IsActive() const { return fActive; }
         Bool_t  IsFolder() const { return kTRUE; }
   virtual void  ls(Option_t *option="*") const;  // *MENU*
           void  SetActive(Bool_t active=kTRUE) { fActive = acti
ve; } // *TOGGLE*
           void  SetBreakin(Int_t breakin=1) { fBreakin = breaki
n; } // *TOGGLE*
           void  SetBreakout(Int_t breakout=1) { fBreakout = bre
akout; } // *TOGGLE*
   TList        *GetListOfTasks() const { return fTasks; }
```

code

```
//The Root browser's picture below has been generated by executi
ng the following script:
{
   TTask *aliroot  = new TTask("aliroot","ALICE reconstruction m
ain task");
   TTask *geominit = new TTask("geomInit","Initialize ALICE geom
etry");
   TTask *matinit  = new TTask("matInit","Initialize ALICE mater
```

```
ials");
   TTask *physinit = new TTask("physInit","Initialize Physics pr
ocesses");
   TTask *tracker  = new TTask("tracker","Track reconstruction m
anager");
   TTask *tpcrec   = new TTask("tpcrec","TPC reconstruction");
   TTask *itsrec   = new TTask("itsrec","ITS reconstruction");
   TTask *muonrec  = new TTask("muonRec","Muon Reconstruction");
   TTask *phosrec  = new TTask("phosRec","Phos Reconstruction");
   TTask *richrec  = new TTask("richRec","Rich Reconstruction");
   TTask *trdrec   = new TTask("trdRec","TRD Reconstruction");
   TTask *globrec  = new TTask("globRec","Global Track Reconstru
ction");
   TTask *pstats   = new TTask("printStats","Print Run Statistic
s");
   TTask *run      = new TTask("run","Process one run");
   TTask *event    = new TTask("event","Process one event");
   aliroot->Add(geominit);
   aliroot->Add(matinit);
   aliroot->Add(physinit);
   aliroot->Add(run);
   run->Add(event);
   event->Add(tracker);
   event->Add(muonrec);
   event->Add(phosrec);
   event->Add(richrec);
   event->Add(trdrec);
   event->Add(globrec);
   tracker->Add(tpcrec);
   tracker->Add(itsrec);
   run->Add(pstats);

   gROOT->GetListOfBrowsables()->Add(aliroot,"aliroot");
   new TBrowser;
}
```

example

数据结构

datarecord

datarecord.txt

```
//  Release 3.02.06
//  A ROOT file is mostly a suite of consecutive data records wi
th the following format
// <Name>;<Cycle> uniquely identifies the record in a directory
// ----------TKey-(never compressed)--------------
//  byte 0->3  Nbytes    = Number of bytes in compressed record
(Tkey+data)   TKey::fNbytes
//       4->5  Version   = TKey class version identifier
              TKey::fVersion
//       6->9  ObjLen    = Number of bytes of uncompressed data
              TKey::fObjLen
//      10->13 Datime    = Date and time when record was written
 to file      TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15 KeyLen    = Number of bytes in key structure (TKe
y)            TKey::fKeyLen
//      16->17 Cycle     = Cycle of key (e.g. 1)
              TKey::fCycle
//      18->21 SeekKey   = Byte offset of record itself (consist
ency check)   TKey::fSeekKey
//      22->25 SeekPdir  = Byte offset of parent directory recor
d             TKey::fSeekPdir
//      26->26 lname     = Number of bytes in the class name
              TKey::fClassName
//      27->.. ClassName = Object Class Name
              TKey::fClassName
//       0->0  lname     = Number of bytes in the object name
              TNamed::fName
//       1->.. Name      = lName bytes with the name of the obje
ct            TNamed::fName
//       0->0  lTitle    = Number of bytes in the object title
              TNamed::fTitle
//       1->.. Title     = lTitle bytes with the title of the ob
ject          TNamed::fTitle
// ----------DATA---(may be compressed)-----------
//       0->..             The data object itself.  For an examp
le, see dobject.txt
```

dataobject

dobject.txt

```
//  Release 3.02.06
// Here is the format of a class object in DATA that uses the de
fault streamer.
// Objects of many classes with custom streamers can have very s
imilar formats.
//----------------
//  0->3  ByteCount = Number of remaining bytes in object (uncom
pressed)
//        | OR'd with kByteCountMask (0x40000000)
//  4->.. ClassInfo = Information about class of object
//        | If this is the first occurrence of an object of this
 class in the record
//                       |  4->7  -1        = New class tag (con
stant kNewClassTag = 0xffffffff)
//                       |  8->.. Classname = Object Class Name
(null terminated string)
//        | Otherwise
//        |  4->7 clIdx      = Byte offset of new class tag in r
ecord, plus 2.
//        | OR'd with kClassMask (0x80000000)
//  0->3  ByteCount = Number of remaining bytes in object (uncom
pressed)
//        | OR'd with kByteCountMask (0x40000000)
//  4->5  Version   = Version of Class
//
// The rest consists of objects of base classes and persistent n
on-static data members.
// Data members marked as transient are not stored.
//
//  6->.. Sequentially, Objects of each base class from which th
is class is derived
//    (rarely more than one)
//  0->.. Sequentially, Objects of all non-static persistent dat
```

```
a members.
//
// Class objects are broken down recursively as above.
//
//      Built in types are stored as follows:
// 1 Byte: char, unsigned char
//      2 Bytes: short, unsigned short
//      4 Bytes: int, unsigned int, float
//      8 Bytes: long, unsigned long, double
// Note that a long (signed or unsigned) is stored as 8 bytes ev
en if it is only four bytes
//  in memory.  In that case, it is filled with leading zeros (o
r ones, for a negative value).
//
```

FreeSegments

freesegments.txt

```
//Format of FreeSegments record, release 3.02.06.  It is never c
ompressed.
//It is probably not accessed by its key, but from its offset gi
ven in the file header.
// ----------TKey---------------
//  byte 0->3  Nbytes    = Number of bytes in compressed record
(TKey+data)       TKey::fNbytes
//       4->5  Version   = TKey class version identifier
                  TKey::fVersion
//       6->9  ObjLen    = Number of bytes of uncompressed data
                  TKey::fObjLen
//      10->13 Datime    = Date and time when record was written
 to file          TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15 KeyLen    = Number of bytes in key structure  (TK
ey)               TKey::fKeyLen
//      16->17 Cycle     = Cycle of key
                  TKey::fCycle
//      18->21 SeekKey   = Byte offset of record itself (consist
ency check)       TKey::fSeekKey
//      22->25 SeekPdir  = Byte offset of parent directory recor
d (TFile)         TKey::fSeekPdir
//      26->26 lname     = Number of bytes in the class name (5)
                  TKey::fClassName
//      27->.. ClassName = Object Class Name ("TFile")
                  TKey::fClassName
//       0->0  lname     = Number of bytes in the object name
                  TNamed::fName
//       1->.. Name      = lName bytes with the name of the obje
ct <file name>    TNamed::fName
//       0->0  lTitle    = Number of bytes in the object title
                  TNamed::fTitle
```

```
//       1->.. Title     = lTitle bytes with the title of the ob
ject <file title> TNamed::fTitle
// ----------DATA---------------
//       0->1  Version   = TFree class version identifier
                  TFree::Class_Version()
//       2->5  fFirst    = First free byte of first free segment
                  TFree::fFirst
//       6->9  fLast     = Byte after last free byte of first fr
ee segment        TFree::fLast
//       ....  Sequentially, Version, fFirst and fLast of additi
onal free segments.
//       ....  There is always one free segment beginning at fil
e end and ending before 2000000000.
```

gap

gap.txt

```
//  A gap (free segment in middle of file) has the following for
mat.
// -------------------------
//  byte 0->3  Nbytes    = Negative of number of bytes in gap
//       4->.. irrelevant
```

header

header.txt

```
// Here is the file header format as of release 3.02.06.  It is
never compressed.
// -----------------------------------
// byte  0->3  "root"      = Identifies this file as a ROOT file
.
//       4->7  Version     = File format version
        TFile::fVersion
//                         |  (10000*major+100*minor+cycle (e.g.
 30203 for 3.2.3))
//       8->11 BEGIN       = Byte offset of first data record (6
4)      TFile::fBEGIN
//      12->15 END         = Pointer to first free word at the E
OF      TFile::fEND
//                         | (will be == to file size in bytes)
//      16->19 SeekFree    = Byte offset of FreeSegments record
        TFile::fSeekFree
//      20->23 NbytesFree  = Number of bytes in FreeSegments rec
ord     TFile::fNBytesFree
//      24->27 nfree       = Number of free data records
//      28->31 NbytesName  = Number of bytes in TKey+TNamed for
TFile at creation TDirectory::fNbytesName
//      32->32 Units       = Number of bytes for file pointers (
4)      TFile::fUnits
//      33->36 Compress    = Zip compression level (i.e. 0-9)
        TFile::fCompress
//      37->40 SeekInfo    = Byte offset of StreamerInfo record
        TFile::fSeekInfo
//      41->44 NbytesInfo  = Number of bytes in StreamerInfo rec
ord     TFile::fNbytesInfo
//      45->63             = Unused??
```

KeysList

keyslist.txt

```
//  Format of KeysList record in release 3.02.06.  It is never c
ompressed.
//  There is one KeysList record for the main (TFile) directory
and one per non-empty subdirectory.
//  It is probably not accessed by its key, but from its offset
given in the directory data.
// ----------TKey---------------
//  byte 0->3  Nbytes    = Number of bytes in compressed record
(TKey+data)   TKey::fNbytes
//       4->5  Version   = TKey class version identifier
              TKey::fVersion
//       6->9  ObjLen    = Number of bytes of uncompressed data
              TKey::fObjLen
//      10->13 Datime    = Date and time when record was written
 to file      TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15 KeyLen    = Number of bytes in the key structure
(TKey)        TKey::fKeyLen
//      16->17 Cycle     = Cycle of key
              TKey::fCycle
//      18->21 SeekKey   = Byte offset of record itself (consist
ency check)   TKey::fSeekKey
//      22->25 SeekPdir  = Byte offset of parent directory recor
d (directory) TKey::fSeekPdir
//      26->26 lname     = Number of bytes in the class name (5
or 10)        TKey::fClassName
//      27->.. ClassName = Object Class Name ("TFile" or "TDirec
tory")        TKey::fClassName
//       0->0  lname     = Number of bytes in the object name
              TNamed::fName
//       1->.. Name      = lName bytes with the name of the obje
ct <directory name> TNamed::fName
```

```
//       0->0  lTitle    = Number of bytes in the object title
              TNamed::fTitle
//       1->.. Title     = lTitle bytes with the title of the ob
ject <directory title> TNamed::fTitle
// ----------DATA---------------
//       0->3  NKeys     = Number of keys in list (i.e. records
in directory (non-recursive))
//                       | Excluded:: The directory itself, Keys
List, StreamerInfo, and FreeSegments
//       4->.. TKey      = Sequentially for each record in direc
tory,
//                       |  the entire TKey portion of each reco
rd is replicated.
//                       |  Note that SeekKey locates the record
.
```

StreamerInfo

StreamerInfo.txt

```
//  Format of StreamerInfo record in release 3.02.06.
//  It is probably not accessed by its key, but from its offset
given in the file header.
//  The StreamerInfo record DATA consists of a TList (list) obje
ct containing elements
//  of class TStreamerInfo.
// ----------TKey-(never compressed)----------------------
//  byte 0->3  Nbytes    = Number of bytes in compressed record
(TKey+data)         TKey::fNbytes
//       4->5  Version   = TKey class version identifier
                    TKey::fVersion
//       6->9  ObjLen    = Number of bytes of uncompressed data
                    TKey::fObjLen
//      10->13 Datime    = Date and time when record was written
 to file            TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15 KeyLen    = Number of bytes in key structure (TKe
y) (64)             TKey::fKeyLen
//      16->17 Cycle     = Cycle of key
                    TKey::fCycle
//      18->21 SeekKey   = Byte offset of record itself (consist
ency check)         TKey::fSeekKey
//      22->25 SeekPdir  = Byte offset of parent directory recor
d (TFile)           TKey::fSeekPdir
//      26->26 lname     = Number of bytes in the class name (5)
                    TKey::fClassName
//      27->31 ClassName = Object Class Name ("TList")
                    TKey::fClassName
//      32->32 lname     = Number of bytes in the object name (1
2)                  TNamed::fName
//      33->44 Name      = lName bytes with the name of the obje
ct ("StreamerInfo") TNamed::fName
```

```
//      45->45 lTitle    = Number of bytes in the object title (
18)                 TNamed::fTitle
//      46->63 Title     = lTitle bytes with the title of the ob
ject                TNamed::fTitle
//                       | ("Doubly linked list")
// ----------TList-(always compressed at level 1 (even if compre
ssion level 0))----
// The DATA is a TList collection object containing TStreamerInf
o objects.
// Below is the format of this TList data.
//
// Here is the format of a TList object in Release 3.02.06.
// Comments and offsets refer specifically to its use in the Str
eamerInfo record.
//--------
//      0->3  ByteCount = Number of remaining bytes in TList obj
ect (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->5  Version   = Version of TList Class
//      6->15           = TObject object (a base class of TList)
 (see tobject.txt).
//                      |   Objects in StreamerInfo record are n
ot referenced.
//                      |   Would be two bytes longer (6->17) if
 object were referenced.
//     16->16 fName     = Number of bytes in name of TList objec
t, followed by the
//                      |   name itself.  (TCollection::fName).
 The TList object in
//                      |   StreamerInfo record is unnamed, so b
yte contains 0.
//     17->20 nObjects  = Number of objects in list.
//     21->.. objects   = Sequentially, TStreamerInfo Objects in
 the list.
//                      | In the StreamerInfo record, the object
s in the list are
//                      |   TStreamerInfo objects.  There will b
e one TStreamerInfo
//                      |   object for every class used in data
```

```
records other than
//                      |   core records and the the StreamerInf
o record itself.
//-------
// Here is the format of a TStreamerInfo object in Release 3.02.
06.
// Note: Although TStreamerInfo does not use the default streame
r, it has the same
// format as if it did.  (compare with dobject.txt)
//      0->3  ByteCount = Number of remaining bytes in TStreamer
Info object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->.. ClassInfo = Information about TStreamerInfo class
//                      | If this is the first occurrence of a T
StreamerInfo object in the record
//                      |  4->7  -1        = New class tag (cons
tant kNewClassTag = 0xffffffff)
//                      |  8->21 Classname = Object Class Name "
TStreamerInfo" (null terminated)
//                      | Otherwise
//                      |  4->7 clIdx      = Byte offset of new
class tag in record, plus 2.
//                      | OR'd with kClassMask (0x80000000)
//      0->3  ByteCount = Number of remaining bytes in TStreamer
Info object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->5  Version   = Version of TStreamerInfo Class
// -Begin TNamed object (Base class of TStreamerInfo)
//      6->9  ByteCount = Number of remaining bytes in TNamed ob
ject
//                      |   OR'd with kByteCountMask (0x40000000
)
//     10->11 Version   = Version of TNamed Class
//     12->21           = TObject object (Base class of TNamed)
(see tobject.txt).
//                      |   Objects in StreamerInfo record are n
ot referenced.
//                      |   Would be two bytes longer (12->23) i
```

```
f object were referenced.
//     22->.. fName     = Number of bytes in name of class that
this TStreamerInfo object
//                      |   describes, followed by the class nam
e itself.  (TNamed::fName).
//      0->.. fTitle    = Number of bytes in title of class that
 this TStreamerInfo object
//                      |   describes, followed by the class tit
le itself.  (TNamed::fTitle).
//                      |  (Class title may be zero length)
// -End TNamed object
//      0->3  fCheckSum = Check sum for class that this TStreame
rInfo object describes.
//                      |  This checksum is over all base classe
s and all persistent
//                      |  non-static data members.  It is compu
ted by TClass::GetCheckSum().
//                      |  (TStreamerInfo::fCheckSum)
//      4->7  fClassVersion = Version of class that this TStream
erInfo object describes.
//                      |   (TStreamerInfo::fClassVersion)
// -Begin TObjArray object (Data member of TStreamerInfo)
//      0->3  ByteCount = Number of remaining bytes in TObjArray
 object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->.. ClassInfo = Information about TObjArray class
//                      | If this is the first occurrence of a T
ObjArray object in the record
//                      |  4->7  -1        = New class tag (cons
tant kNewClassTag = 0xffffffff)
//                      |  8->17 Classname = Object Class Name "
TObjArray" (null terminated)
//                      | Otherwise
//                      |  4->7 clIdx      = Byte offset of new
class tag in record, plus 2.
//                      | OR'd with kClassMask (0x80000000)
//      0->3  ByteCount = Number of remaining bytes in TObjArray
 object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
```

```
)
//      4->5  Version   = Version of TObjArray Class
//      6->15           = TObject object (a base class of TObjAr
ray) (see tobject.txt).
//                      |   Objects in StreamerInfo record are n
ot referenced.
//                      |   Would be two bytes longer (6->17) if
 object were referenced.
//     16->16 fName     = Number of bytes in name of TObjArray o
bject, followed by the
//                      |   name itself.  (TCollection::fName).
 TObjArray objects in
//                      |   StreamerInfo record are unnamed, so
byte contains 0.
//     17->20 nObjects  = Number of objects (derived from TStrea
merElement) in array.
//     21->24 fLowerBound = Lower bound of array.  Will always b
e 0 in StreamerInfo record.
//     25->.. objects   = Sequentially, TStreamerElement objects
 in the array.
//                      | In a TStreamerInfo object, the objects
 in the TObjArray are
//                      |   of various types (described below),
all of which inherit
//                      |   directly from TStreamerElement objec
ts.  There will be one
//                      |   such object for every base class of
the class that the
//                      |   TStreamerInfo object describes, and
also one such object for
//                      |   each persistent non-static data memb
er of the class that the
//                      |   TStreamerInfo object describes.
// -End TObjArray object and TStreamerInfo object
//-------
//  The objects stored in the TObjectArray in TStreamerInfo are
of various classes, each of
//     which inherits directly from the TStreamerElement class.
 The possible classes (which
//     we refer to collectively as TStreamer<XXX>) are:
```

```
//
//  TStreamerBase:          Used for a base class.  All others b
elow used for data members.
//  TStreamerBasicType:     For a basic type
//  TStreamerString:        For type TString
//  TStreamerBasicPointer:  For pointer to array of basic types
//  TStreamerObject:        For an object derived from TObject
//  TStreamerObjectPointer: For pointer to an object derived fro
m TObject
//  TStreamerLoop:          For pointer to an array of objects
//  TStreamerObjectAny:     For an object not derived from TObje
ct
//  TStreamerSTL:           For an STL container (not yet used??
)
//  TStreamerSTLString:     For an STL string (not yet used??)
//-------
// Here is the format of a TStreamer<XXX> object in Release 3.02
.06.
// In description below,
//     0->3  ByteCount = Number of remaining bytes in TStreamer<
XXX> object (uncompressed)
//                     |   OR'd with kByteCountMask (0x40000000)
//     4->.. ClassInfo = Information about the specific TStreame
r<XXX> class
//                     | If this is the first occurrence of a TS
treamerXXX object in the record
//                     |  4->7  -1        = New class tag (const
ant kNewClassTag = 0xffffffff)
//                     |  8->.. Classname = Object Class Name "T
Streamer<XXX>" (null terminated)
//                     | Otherwise
//                     |  4->7 clIdx      = Byte offset of new c
lass tag in record, plus 2.
//                    | OR'd with kClassMask (0x80000000)
//     0->3  ByteCount = Number of remaining bytes in TStreamer<
XXX> object (uncompressed)
//                   |   OR'd with kByteCountMask (0x40000000)
//     4->5  Version   = Version of TStreamer<XXX> Class
// -Begin TStreamerElement object (Base class of TStreamerXXX)
//     0->3  ByteCount = Number of remaining bytes in TStreamerE
```

```
lement object (uncompressed)
//                    |   OR'd with kByteCountMask (0x40000000)
//     4->5  Version   = Version of TStreamerElement Class
// -Begin TNamed object (Base class of TStreamerElement)
//     6->9  ByteCount = Number of remaining bytes in TNamed obj
ect
//                      |   OR'd with kByteCountMask (0x40000000)
//     10->11 Version   = Version of TNamed Class
//     12->21           = TObject object (Base class of TNamed)
(see tobject.txt).
//                       |   Objects in StreamerInfo record are n
ot referenced.
//                       |   Would be two bytes longer (12->23) i
f object were referenced.
//     22->.. fName     = Number of bytes in class name of base
class or member name of
//                       | data member that this TStreamerElement
 object describes,
//                       | followed by the name itself. (TNamed::
fName).
//      0->.. fTitle    = Number of bytes in title of base class
 or data member that this
//                       | TStreamerElement object describes, fol
lowed by the title itself.
//                       |  (TNamed::fTitle).
// -End TNamed object
//      0->3  fType     = Type of data described by this TStream
erElement.
//                       |   (TStreamerElement::fType)
//                       |   Built in types:
//                       |   1:char, 2:short, 3:int, 4:long, 5:fl
oat, 8:double
//                       |   11, 12, 13, 14:unsigned char, short,
 int, long respectively
//                       |   6: an array dimension (counter)
//                       |   15: bit mask (used for fBits field)
//                       |
//                       |   Pointers to built in types:
//                       |   40 + fType of built in type (e.g. 43
: pointer to int)
```

```
//                      |
//                      |   Objects:
//                      |   65:TString, 66:TObject, 67:TNamed
//                      |   0: base class (other than TObject or
 TNamed)
//                      |   61: object data member derived from
TObject (other than TObject or TNamed)
//                      |   62: object data member not derived f
rom TObject
//                      |   63: pointer to object derived from T
Object (pointer can't be null)
//                      |   64: pointer to object derived from T
Object (pointer may be null)
//                      |   501: pointer to an array of objects
//                      |   500: an STL string or container
//                      |
//                      |   Arrays:
//                      |   20 + fType of array element (e.g. 23
: array of int)
//                      |
//      4->7  fSize     = Size of built in type or of pointer to
 built in type. 0 otherwise.
//                      |  (TStreamerElement::fSize).
//      8->11 fArrayLength = Size of array (0 if not array)
//                      |  (TStreamerElement::fArrayLength).
//     12->15 fArrayDim = Number of dimensions of array (0 if no
t an array)
//                      |  (TStreamerElement::fArrayDim).
//     16->35 fMaxIndex = Five integers giving the array dimensi
ons (0 if not applicable)
//                      |  (TStreamerElement::fMaxIndex).
//     36->.. fTypeName = Number of bytes in name of the data ty
pe of the data member that
//                      |  the TStreamerElement object describes
, followed by the name
//                      |  itself.  If this TStreamerElement obj
ect defines a base class
//                      |  rather than a data member, the name u
sed is 'BASE'.
//                      |  (TStreamerElement::fTypeName).
```

```
// -End TStreamerElement object
//     The remaining data is specific to the type of TStreamer<X
XX> class.
//      For TStreamerInfoBase:
//      0->3  fBaseVersion   = Version of base class that this T
StreamerElement describes.
//      For TStreamerBasicType:
//            No specific data
//      For TStreamerString:
//            No specific data
//      For TStreamerBasicPointer:
//      0->3  fCountVersion = Version of class with the count (a
rray dimension)
//      4->.. fCountName= Number of bytes in the name of the dat
a member holding
//                      | the count, followed by the name itself
.
//      0->.. fCountName= Number of bytes in the name of the cla
ss holding the
//                      | count, followed by the name itself.
//      For TStreamerObject:
//            No specific data
//      For TStreamerObjectPointer:
//            No specific data
//      For TStreamerLoop:
//      0->3  fCountVersion = Version of class with the count (a
rray dimension)
//      4->.. fCountName= Number of bytes in the name of the dat
a member holding
//                      | the count, followed by the name itself
.
//      0->.. fCountClass= Number of bytes in the name of the cl
ass holding the
//                      | count, followed by the name itself.
//      For TStreamerObjectAny:
//            No specific data
//      For TStreamerSTL:
//      0->3  fSTLtype  = Type of STL container:
//                      | 1:vector, 2:list, 3:deque, 4:map, 5:se
t, 6:multimap, 7:multiset
```

```
//      4->7  fCType    = Type contained in STL container:
//                      | Same values as for fType above, with o
ne addition: 365:STL string
//      For TStreamerSTLString:
//            No specific data
```

TClonesArray

TClonesArray.txt

```
// -Here is the format (release 3.02.06)  of the DATA for a TClo
nesArray object in a ROOTIO file.
//      0->3  ByteCount = Number of remaining bytes in TClonesAr
ray object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->.. ClassInfo = Information about TClonesArray class
//                      | If this is the first occurrence of a T
ClonesArray object in the record
//                      |  4->7  -1        = New class tag (cons
tant kNewClassTag = 0xffffffff)
//                      |  8->17 Classname = Object Class Name "
TClonesArray" (null terminated)
//                      | Otherwise
//                      |  4->7 clIdx      = Byte offset of new
class tag in record, plus 2.
//                      | OR'd with kClassMask (0x80000000)
//      0->3  ByteCount = Number of remaining bytes in TClonesAr
ray object (uncompressed)
//                      |   OR'd with kByteCountMask (0x40000000
)
//      4->5  Version   = Version of TClonesArray Class
//      6->15           = TObject object (a base class of TClone
sArray) (see tobject.txt).
//                      |   Would be two bytes longer (6->17) if
 object were referenced.
//     16->.. fName     = Number of bytes in name of TClonesArra
y object, followed by the
//                      |   name itself.  (TCollection::fName).
 This name will be the
//                      |   class name of the cloned object, app
ended with an 's'
//                      |   (e.g. "TXxxs")
```

```
//      0->..           = Number of bytes in name and version of
 the cloned class, followed
//                      |   by the name and version themselves (
e.g. "TXxx;1")
//      0->3  nObjects  = Number of objects in clones array.
//      4->7  fLowerBound= Lower bound of clones array.
//      8->.. objects   = Sequentially, objects in the clones ar
ray.  However, the data
//                      |   ordering depends on whether or not k
BypassStreamer (0x1000) is
//                      |   set in TObject::fBits.   By default,
 it is set.  If it is not set,
//                      |   the objects are streamed sequentiall
y using the streamer of the
//                      |   cloned class (e.g. TXxx::Streamer())
.
//                      |
//                      |   If it is set, the cloned class is sp
lit into its base classes and
//                      |   persistent data members, and those s
treamers are used.  So, if the
//                      |   base classes and persistent data mem
bers of class TXxx are TXxxbase,
//                      |   TXxxdata0, TXxxdata1, etc.,  all the
 TXxxbase data from the entire
//                      |   clones array is streamed first, foll
owed by all the TXxxdata0 data,
//                      |   etc.  This breakdown is not recursiv
e, in that the member objects
//                      |   are not again split.
// -End TClonesArray object
```

TDirectory

TDirectory.txt

```
//  Format of a TDirectory record in release 3.02.06.  It is nev
er compressed.
// ----------TKey--------------
//  byte 0->3  Nbytes    = Number of bytes in compressed record
(Tkey+data)           TKey::fNbytes
//       4->5  Version   = TKey class version identifier
                        TKey::fVersion
//       6->9  ObjLen    = Number of bytes of uncompressed data
                        TKey::fObjLen
//      10->13 Datime    = Date and time when record was written
 to file                TKey::fDatime
//                          | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15 KeyLen    = Number of bytes in key structure (TKe
y)                      TKey::fKeyLen
//      16->17 Cycle     = Cycle of key
                        TKey::fCycle
//      18->21 SeekKey   = Byte offset of record itself (consist
ency check)             TKey::fSeekKey
//      22->25 SeekPdir  = Byte offset of parent directory recor
d                       TKey::fSeekPdir
//      26->26 lname     = Number of bytes in the class name (10
)                       TKey::fClassName
//      27->.. ClassName = Object Class Name ("TDirectory")
                        TKey::fClassName
//       0->0  lname     = Number of bytes in the object name
                        TNamed::fName
//       1->.. Name      = lName bytes with the name of the obje
ct <directory name>     TNamed::fName
//       0->0  lTitle    = Number of bytes in the object title
                        TNamed::fTitle
//       1->.. Title     = lTitle bytes with the title of the ob
ject <direcory title> TNamed::fTitle
```

```
// --------DATA----------------
//       0->0  Modified  = True if directory has been modified
                      TDirectory::fModified
//       1->1  Writable = True if directory is writable
                      TDirectory::fWriteable
//       2->5  DatimeC   = Date and time when directory was crea
ted                   TDirectory::fDatimeC
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//       6->9  DatimeM   = Date and time when directory was last
 modified             TDirectory::fDatimeM
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      10->13 NbytesKeys= Number of bytes in the associated Key
sList record          TDirectory::fNbyteskeys
//      14->17 NbytesName= Number of bytes in TKey+TNamed at cre
ation                 TDirectory::fNbytesName
//      18->21 SeekDir   = Byte offset of directory record in fi
le                    TDirectory::fSeekDir
//      22->25 SeekParent= Byte offset of parent directory recor
d in file             TDirectory::fSeekParent
//      26->29 SeekKeys  = Byte offset of associated KeysList re
cord in file          TDirectory::fSeekKeys
```

TFile

TFile.txt

```
// A ROOT file is a suite of consecutive data records (TKey's) w
ith
// the following format (see also the TKey class). If the key is
// located past the 32 bit file limit (> 2 GB) then some fields
will
// be 8 instead of 4 bytes:
// ----------TKey---------------
//  byte 0->3           Nbytes    = Number of bytes compressed r
ecord (TKey+data)          TKey::fNbytes
//       4->5           Version   = TKey class version identifie
r                          TKey::fVersion
//       6->9           ObjLen    = Number of bytes of uncompres
sed data                   TKey::fObjLen
//      10->13          Datime    = Date and time when record wa
s written to file          TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      14->15          KeyLen    = Number of bytes in key struc
ture (TKey)                TKey::fKeyLen
//      16->17          Cycle     = Cycle of key
                           TKey::fCycle
//      18->21 [18->25] SeekKey   = Byte offset of record itself
 (consistency check) (64)  TKey::fSeekKey
//      22->25 [26->33] SeekPdir  = Byte offset of parent direct
ory record (0)             TKey::fSeekPdir
//      26->26 [34->34] lname     = Number of bytes in the class
 name (5)                  TKey::fClassName
//      27->.. [35->..] ClassName = Object Class Name ("TFile")
                           TKey::fClassName
//       0->0  lname     = Number of bytes in the object name
                           TNamed::fName
//       1->..           Name      = lName bytes with the name of
 the object <file name>    TNamed::fName
```

```
//       0->0           lTitle    = Number of bytes in the objec
t title                    TNamed::fTitle
//       1->..          Title     = lTitle bytes with the title
of the object <file title> TNamed::fTitle
// --------DATA-----------------
//       0->0           lname     = Number of bytes in the TFile
 name                      TNamed::fName
//       1->.. Name      = lName bytes with the name of the TFil
e <file name>              TNamed::fName
//       0->0           lTitle    = Number of bytes in the TFile
 title                     TNamed::fTitle
//       1->..          Title     = lTitle bytes with the title
of the TFile <file title>  TNamed::fTitle
//       0->0           Modified  = True if directory has been m
odified                    TDirectory::fModified
//       1->1           Writable = True if directory is writable
                           TDirectory::fWriteable
//       2->5           DatimeC   = Date and time when directory
 was created               TDirectory::fDatimeC
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//       6->9           DatimeM   = Date and time when directory
 was last modified         TDirectory::fDatimeM
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//      10->13          NbytesKeys= Number of bytes in the assoc
iated KeysList record      TDirectory::fNbyteskeys
//      14->17          NbytesName= Number of bytes in TKey+TNam
ed at creation             TDirectory::fNbytesName
//      18->21 [18->25] SeekDir   = Byte offset of directory rec
ord in file (64)           TDirectory::fSeekDir
//      22->25 [26->33] SeekParent= Byte offset of parent direct
ory record in file (0)     TDirectory::fSeekParent
//      26->29 [34->41] SeekKeys  = Byte offset of associated Ke
ysList record in file      TDirectory::fSeekKeys
```

TObject

TObject.txt

```
// Here is the format of the DATA for a TObject object in Releas
e 3.02.06.
//--------
//  0->1  Version   = Version of TObject Class
//  2->5  fUniqueID = Unique ID of object.  Currently, unless th
is object is or was
//                       | referenced by a TRef or TRefArray, or
 is itself a TRef or TRefArray,
//                       | this field is not used by ROOT.
//  6->9  fBits     = A 32 bit mask containing status bits for t
he object.
//                       | The bits relevant to ROOTIO are:
//                       | 0x00000001 - if object in a list can
be deleted.
//                       | 0x00000008 - if other objects may nee
d to be deleted when this one is.
//                       | 0x00000010 - if object is referenced
by pointer to persistent object.
//                       | 0x00002000 - if object ctor succeeded
 but object shouldn't be used
//                       | 0x01000000 - if object is on Heap.
//                       | 0x02000000 - if object has not been d
eleted.
// The "pidf" field below is present only if this TObject object
 (or an object inheriting
//      from it) is referenced by a pointer to persistent object
.
// 10->11 pidf  = An identifier of the TProcessID record for the
 process that wrote the
//                       | object. This identifier is an unsigne
d short.  The relevant record
//                       | has a name that is the string "Proces
sID" concatenated with the ASCII
```

```
//                        | decimal representation of "pidf" (no
leading zeros).  0 is a valid pidf.
//-------
// No object in the StreamerInfo record will be a reference or r
eferenced, and all objects
//      are on the heap.  So, for each occurrence in the Streame
rInfo record, fUniqueID will be 0,
//      fBits will be 0x03000000, and pidf will be absent.
```

TProcessID

TProcessID.txt

```
// Format of TProcessID record in release 3.02.06.
// Will be present if there are any referenced objects.
// ----------TKey---------------
//   byte 0->3  Nbytes   = Number of bytes in compressed record
(Tkey+data)   TKey::fNbytes
//        4->5  Version  = TKey class version identifier
              TKey::fVersion
//        6->9  ObjLen   = Number of bytes of uncompressed data
              TKey::fObjLen
//       10->13 Datime   = Date and time when record was written
 to file      TKey::fDatime
//                       | (year-1995)<<26|month<<22|day<<17|hou
r<<12|minute<<6|second
//       14->15 KeyLen   = Number of bytes in key structure (TKe
y)            TKey::fKeyLen
//       16->17 Cycle    = Cycle of key
              TKey::fCycle
//       18->21 SeekKey  = Byte offset of record itself (consist
ency check)   TKey::fSeekKey
//       22->25 SeekPdir = Byte offset of parent directory recor
d             TKey::fSeekPdir
//       26->26 lname    = Number of bytes in the class name (10
)             TKey::fClassName
//       27->36 ClassName= Object Class Name ("TProcessID")
              TKey::fClassName
//       37->37 lname    = Number of bytes in the object name
              TNamed::fName
//       38->.. Name     = lName bytes with the name of the obje
ct            TNamed::fName
//                       | (e.g. "ProcessID0")
//        0->0  lTitle   = Number of bytes in the object title
              TNamed::fTitle
//        1->.. Title    = lTitle bytes with the title of the ob
```

```
ject          TNamed::fTitle
//                       | (Identifies processor, time stamp, et
c.)
//                       | See detailed explanation below.
// ----------DATA--------------
//        0->3 ByteCount = Number of remaining bytes in TProcess
ID object (uncompressed)
//                       |   OR'd with kByteCountMask (0x4000000
0)
//        4->5 Version   = Version of TProcessID Class
// -Begin TNamed object (Base class of TProcessID)
//        6->9 ByteCount = Number of remaining bytes in TNamed o
bject (uncompressed)
//                       |   OR'd with kByteCountMask (0x4000000
0)
//       10->11 Version  = Version of TNamed Class
//       12->21          = TObject object (Base class of TNamed)
 (see tobject.txt).
//                       |   The TProcessID object is not itself
 referenced.
//       22->22 lname    = Number of bytes in the object name
              TNamed::fName
//       23->.. Name     = lName bytes with the name of the obje
ct            TNamed::fName
//                       | The name will be "ProcessID" concaten
ated with
//                       | a decimal integer, or "pidf".
//        0->0  lTitle   = Number of bytes in the object title
              TNamed::fTitle
//        1->.. Title    = lTitle bytes with the title of the ob
ject          TNamed::fTitle
//                       | (Identifies processor, time stamp, et
c.)
//                       | See detailed explanation below.
// -End TNamed object
//
//     ----------Explanation of the title of a TProcessID object
-----------------
// The title of a TProcessID object is a globally unique identif
ier of the
```

```
// ROOTIO process that created it.  It is derived from the follo
wing quantities.
//
// 1) The creation time ("fTime) of the TProcessID record.  This
 is a 60 bit time
// in 100ns ticks since Oct. 15, 1582.
//
// 2) A 16 bit random unsigned integer ("clockeq") generated fro
m a seed that is the
//      job's process ID.  The highest two bits are not used.
//
// 3) A six byte unsigned quantity ("fNode") identifying the mac
hine.  If the machine has a
// valid network address, the first four bytes are set to that a
ddress, and the last two bytes
// are stuffed with 0xbe and 0xef respectively.  Otherwise a six
 byte quantity is generated
// from the time and random machine statistics. In this case, th
e high order bit of the
// first byte is set to 1, to distinguish it from a network ID,
where the bytes can be
// no larger than 255.
//
// We the define the following quantities (class TUUID):
        UInt_t    fTimeLow;               // 60 bit time, lowest
 32 bits
        UShort_t  fTimeMid;               // 60 bit time, middle
 16 bits
        UShort_t  fTimeHiAndVersion;      // 60 bit time, highes
t 12 time bits (low 12 bits)
                                          // + 4 UUID version bi
ts (high 4 bits)
                                          // version is 1 if mac
hine has valid network address
                                          // and 3 otherwise.
        UChar_t   fClockSeqHiAndReserved; // high 6 clockseq bit
s (low 6 bits)
                                          // + 2 high bits reser
ved (currently set to binary 10)
        UChar_t   fClockSeqLow;           // low 8 clockseq bits
```

```
        UChar_t   fNode[6];               // 6 node (machine) id
 bytes

// Then the following sprintf() call defines the format of the t
itle string:
   sprintf(Title, "%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02x%02x%
02x",
           fTimeLow, fTimeMid, fTimeHiAndVersion, fClockSeqHiAnd
Reserved,
           fClockSeqLow, fNode[0], fNode[1], fNode[2], fNode[3],
 fNode[4],
           fNode[5]);
// Since the title written to disk is preceded by its byte count
, the delimiting null is not written.
```

TRefArray

TRefArray.txt

```
// Here is the format of the DATA for a TRefArray object in Rele
ase 3.02.06.
//--------
//       0->3  ByteCount = Number of remaining bytes in TRefArra
y object (uncompressed)
//                       |   OR'd with kByteCountMask (0x4000000
0)
//       4->5  Version   = Version of TRefArray Class
//       6->15           = TObject object (Base class of TRefArr
ay) (see tobject.txt).
//                       |   Will be two bytes longer (6->17) if
 TRefArray object is
//                       |   itself referenced (unlikely).
//      16->.. fName     = Number of bytes in name of TRefArray
object, followed by the
//                       |   name itself.  (TCollection::fName).
 Currently, TRefArrays
//                       |   are not named, so this is a single
byte containing 0.
//       0->3  nObjects  | Number of object references (fUIDs) i
n this TRefArray.
//       4->7  fLowerBound= Lower bound of array.  Typically 0.
//       8->9  pidf  = An identifier of the TProcessID record fo
r the process that wrote the
//                       | referenced objects. This identifier i
s an unsigned short.  The relevant
//                       | record has a name that is the string
"ProcessID" concatenated with the
//                       | ASCII decimal representation of "pidf
" (no leading zeros).
//                       | 0 is a valid pidf.
//      10->.. fUIDs     = Sequentially, object Unique ID's.
//                       | Each Unique ID is a four byte unsigne
```

```
d integer.
//                        | If non-zero, it matches the Unique ID
 in the referenced
//                        | object.  If zero, it is an unused ele
ment in the array.
//                        | The fUIDs are written out only up to
the last used element,
//                        | so the last fUID will always be non-z
ero.
```

TRef

TRef.txt

```
// Here is the format of the DATA for a TRef object in Release 3
.02.06.
//--------
//  0->1  Version   = Version of TObject Class (base class of TR
ef)
//  2->5  fUniqueID = Unique ID of referenced object.  Typically
, every referenced
//                        | object has an ID that is a positive i
nteger set to a counter
//        | of the number of referenced objects in the file, beg
inning at 1.
//                        | fUniqueID in the TRef object matches
fUniqueID in the
//                        | referenced object.
//  6->9  fBits     = A 32 bit mask containing status bits for t
he TRef object.
//                        | The bits relevant to ROOTIO are:
//        | 0x00000008 - Other objects may need to be deleted wh
en this one is.
//        | 0x00000010 - Object is referenced by pointer to pers
istent object.
//        | 0x01000000 - Object is on Heap.
//        | 0x02000000 - Object has not been deleted.
// 10->11 pidf  = An identifier of the TProcessID record for the
 process that wrote the
//                        | referenced object. This identifier is
 an unsigned short.  The relevant
//                        | record has a name that is the string
"ProcessID" concatenated with the
//                        | ASCII decimal representation of "pidf
" (no leading zeros).
//                        | 0 is a valid pidf.
//-------
```

TTree

TTree.txt

```
Here is the streamer information for TTree related classes in re
lease 3.02.06:
(For the explanation of the meaning of the type, see "fType" in
"streamerinfo.txt".)
-----------------------------------------------------
StreamerInfo for class: TTree, version=6
  BASE          TNamed          offset=  0 type=67 The basis for
 a named object (name, title)
  BASE          TAttLine        offset=  0 type= 0 Line attribut
es
  BASE          TAttFill        offset=  0 type= 0 Fill area att
ributes
  BASE          TAttMarker      offset=  0 type= 0 Marker attrib
utes
  Stat_t        fEntries        offset=  0 type= 8 Number of ent
ries
  Stat_t        fTotBytes       offset=  0 type= 8 Total number
of bytes in all branches before compression
  Stat_t        fZipBytes       offset=  0 type= 8 Total number
of bytes in all branches after compression
  Stat_t        fSavedBytes     offset=  0 type= 8 Number of aut
osaved bytes
  Int_t         fTimerInterval  offset=  0 type= 3 Timer interva
l in milliseconds
  Int_t         fScanField      offset=  0 type= 3 Number of run
s before prompting in Scan
  Int_t         fUpdate         offset=  0 type= 3 Update freque
ncy for EntryLoop
  Int_t         fMaxEntryLoop   offset=  0 type= 3 Maximum numbe
r of entries to process
  Int_t         fMaxVirtualSize offset=  0 type= 3 Maximum total
 size of buffers kept in memory
  Int_t         fAutoSave       offset=  0 type= 3 Autosave tree
```

```
 when fAutoSave bytes produced
  Int_t         fEstimate       offset=  0 type= 3 Number of ent
ries to estimate histogram limits
  TObjArray     fBranches       offset=  0 type=61 List of Branc
hes
  TObjArray     fLeaves         offset=  0 type=61 Direct pointe
rs to individual branch leaves
  TArrayD       fIndexValues    offset=  0 type=62 Sorted index
values
  TArrayI       fIndex          offset=  0 type=62 Index of sort
ed values
  TList*        fFriends        offset=  0 type=64 pointer to li
st of friend elements

StreamerInfo for class: TAttLine, version=1
  Color_t       fLineColor      offset=  0 type= 2 line color
  Style_t       fLineStyle      offset=  0 type= 2 line style
  Width_t       fLineWidth      offset=  0 type= 2 line width

StreamerInfo for class: TAttFill, version=1
  Color_t       fFillColor      offset=  0 type= 2 fill area col
or
  Style_t       fFillStyle      offset=  0 type= 2 fill area sty
le

StreamerInfo for class: TAttMarker, version=1
  Color_t       fMarkerColor    offset=  0 type= 2 Marker color
index
  Style_t       fMarkerStyle    offset=  0 type= 2 Marker style
  Size_t        fMarkerSize     offset=  0 type= 5 Marker size

StreamerInfo for class: TBranch, version=7
  BASE          TNamed          offset=  0 type=67 The basis for
 a named object (name, title)
  Int_t         fCompress       offset=  0 type= 3 (=1 branch is
 compressed, 0 otherwise)
  Int_t         fBasketSize     offset=  0 type= 3 Initial Size
of  Basket Buffer
  Int_t         fEntryOffsetLen offset=  0 type= 3 Initial Lengt
h of fEntryOffset table in the basket buffers
```

```
  Int_t         fWriteBasket    offset=  0 type= 3 Last basket n
umber written
  Int_t         fEntryNumber    offset=  0 type= 3 Current entry
 number (last one filled in this branch)
  Int_t         fOffset         offset=  0 type= 3 Offset of thi
s branch
  Int_t         fMaxBaskets     offset=  0 type= 6 Maximum numbe
r of Baskets so far
  Int_t         fSplitLevel     offset=  0 type= 3 Branch split
level
  Stat_t        fEntries        offset=  0 type= 8 Number of ent
ries
  Stat_t        fTotBytes       offset=  0 type= 8 Total number
of bytes in all leaves before compression
  Stat_t        fZipBytes       offset=  0 type= 8 Total number
of bytes in all leaves after compression
  TObjArray     fBranches       offset=  0 type=61 -> List of Br
anches of this branch
  TObjArray     fLeaves         offset=  0 type=61 -> List of le
aves of this branch
  TObjArray     fBaskets        offset=  0 type=61 -> List of ba
skets of this branch
  Int_t*        fBasketBytes    offset=  0 type=43 [fMaxBaskets]
 Length of baskets on file
  Int_t*        fBasketEntry    offset=  0 type=43 [fMaxBaskets]
 Table of first entry in eack basket
  Seek_t*       fBasketSeek     offset=  0 type=43 [fMaxBaskets]
 Addresses of baskets on file
  TString       fFileName       offset=  0 type=65 Name of file
where buffers are stored ("" if in same file as Tree header)

StreamerInfo for class: TBranchElement, version=7
  BASE          TBranch         offset=  0 type= 0 Branch descri
ptor
  TString       fClassName      offset=  0 type=65 Class name of
 referenced object
  TString       fParentName     offset=  0 type=65 Name of paren
t class
  TString       fClonesName     offset=  0 type=65 Name of class
 in TClonesArray (if any)
```

```
  Int_t         fClassVersion   offset=  0 type= 3 Version numbe
r of class
  Int_t         fID             offset=  0 type= 3 element seria
l number in fInfo
  Int_t         fType           offset=  0 type= 3 branch type
  Int_t         fStreamerType   offset=  0 type= 3 branch stream
er type
  Int_t         fMaximum        offset=  0 type= 3 Maximum entri
es for a TClonesArray or variable array
  TBranchElement*fBranchCount    offset=  0 type=64 pointer to p
rimary branchcount branch
  TBranchElement*fBranchCount2   offset=  0 type=64 pointer to s
econdary branchcount branch

StreamerInfo for class: TLeaf, version=2
  BASE          TNamed          offset=  0 type=67 The basis for
 a named object (name, title)
  Int_t         fLen            offset=  0 type= 3 Number of fix
ed length elements
  Int_t         fLenType        offset=  0 type= 3 Number of byt
es for this data type
  Int_t         fOffset         offset=  0 type= 3 Offset in Clo
nesArray object (if one)
  Bool_t        fIsRange        offset=  0 type=11 (=kTRUE if le
af has a range, kFALSE otherwise)
  Bool_t        fIsUnsigned     offset=  0 type=11 (=kTRUE if un
signed, kFALSE otherwise)
  TLeaf*        fLeafCount      offset=  0 type=64 Pointer to Le
af count if variable length

StreamerInfo for class: TLeafElement, version=1
  BASE          TLeaf           offset=  0 type= 0 Leaf: descrip
tion of a Branch data type
  Int_t         fID             offset=  0 type= 3 element seria
l number in fInfo
  Int_t         fType           offset=  0 type= 3 leaf type
```

预定义

RVersion

RVersion.h

```
#ifndef ROOT_RVersion
#define ROOT_RVersion

/* Version information automatically generated by installer. */

/*
 * These macros can be used in the following way:
 *
 *    #if ROOT_VERSION_CODE >= ROOT_VERSION(2,23,4)
 *       #include <newheader.h>
 *    #else
 *       #include <oldheader.h>
 *    #endif
 *
*/

#define ROOT_RELEASE "6.06/02"
#define ROOT_RELEASE_DATE "Mar  3 2016"
#define ROOT_RELEASE_TIME "10:36:03"
#define ROOT_VERSION(a,b,c) (((a) << 16) + ((b) << 8) + (c))
#define ROOT_VERSION_CODE ROOT_VERSION(6,6,2) /* 394754 */

#endif
```

文件更新版本

- Beautify 6.06.02
- DataAnalysis 6.06.02
- file 6.06.02
- math 6.06.02
- mlp 6.06.02
- net
- picture 6.06.02
- roofit
- Spectrum 6.06.02
- thread
- time
- tmva
- TRandom 6.06.02
- TSelector 6.06.02
- system 6.06.02

- RVersion.h 6.06.02

- TDirectory.cxx 6.06.02
- TDirectory.h 6.06.02
- TDirectory.txt 6.06.02
- TDirectoryFile.cxx 6.06.02
- TDirectoryFile.h 6.06.02
- TEllipse.h 6.06.02
- TEllipse.cxx 6.06.02
- TNamed.cxx 6.06.02
- TNamed.h 6.06.02
- TObject.cxx 6.06.02
- TObject.h 6.06.02
- TObject.txt 6.06.02
- TString.cxx 6.06.02
- TString.h 6.06.02
- TTask.cxx 6.06.02

- TTask.h 6.06.02

文件夹分类

- Beautify/
  - TAttAxis.cxx
  - TAttAxis.h
  - TAxis.cxx
  - TAxis.h
  - TCanvas.cxx
  - TCanvas.h*
  - TGaxis.cxx
  - TGaxis.h
  - TLatex.cxx
  - TLatex.h
  - TLegend.cxx
  - TLegendEntry.cxx
  - TLegendEntry.h
  - TLegend.h
  - TLine.cxx
  - TLine.h
  - TPad.cxx
  - TPad.h
  - TPaveStats.cxx
  - TPaveStats.h
  - TPavesText.cxx
  - TPaveText.cxx
  - TPaveText.h
  - TText.cxx
  - TText.h
- DataAnalysis/
  - TCut.cxx
  - TCutG.cxx
  - TCutG.h
  - TCut.h

- file/
  - TBranch.cxx
  - TBranch.h
  - TChain.cxx
  - TChain.h
  - TEventList.cxx
  - TEventList.h
  - TFile.cxx
  - TFile.h
  - TList.cxx
  - TList.h
  - TNtuple.cxx
  - TNtupleD.cxx
  - TNtupleD.h
  - TNtuple.h
  - TTree.cxx
  - TTree.h
  - TTreePlayer.cxx
  - TTreePlayer.h
- Fit/
  - TBackCompFitter.cxx
  - TBackCompFitter.h
  - TFitter.cxx
  - TFitter.h
  - TFractionFitter.cxx
  - TFractionFitter.h
  - TLinearFitter.cxx
  - TLinearFitter.h
  - TMinuit2TraceObject.cxx
  - TMinuit2TraceObject.h
  - TMinuit.cxx
  - TMinuit.h
  - TMinuitMinimizer.cxx
  - TMinuitMinimizer.h
  - TVirtualFitter.cxx
  - TVirtualFitter.h

- gui/
- math/
  - TMathBase.cxx
  - TMathBase.h
  - TMath.cxx
  - TMath.h
  - TMatrixDBasefwd.h
  - TMatrixDBase.h
  - TMatrixDfwd.h
  - TMatrixD.h
  - TMatrixDSparse.h
  - TMatrixDSym.h
  - TMatrix.h
  - TMatrixTBase.cxx
  - TMatrixTBase.h
  - TMatrixT.cxx
  - TMatrixT.h
  - TMatrixTSparse.cxx
  - TMatrixTSparse.h
  - TMatrixTSym.cxx
  - TMatrixTSym.h
  - TVector2.cxx
  - TVector2.h
  - TVector3.cxx
  - TVector3.h
  - TVectorD.h
  - TVectorT.cxx
  - TVectorT.h
- mlp/
  - TMLPAnalyzer.cxx
  - TMLPAnalyzer.h
  - TMultiLayerPerceptron.cxx
  - TMultiLayerPerceptron.h
  - TNeuron.cxx
  - TNeuron.h

- net/
  - MessageTypes.h
  - TBuffer.cxx
  - TBufferFile.cxx
  - TBufferFile.h
  - TBuffer.h
  - TFileCacheWrite.cxx
  - TFileCacheWrite.h
  - TFileMerger.cxx
  - TFileMerger.h
  - TMemFile.cxx
  - TMemFile.h
  - TMessage.cxx
  - TMessage.h
  - TMonitor.cxx
  - TMonitor.h
  - TPServerSocket.cxx
  - TPServerSocket.h
  - TPSocket.cxx
  - TPSocket.h
  - TServerSocket.cxx
  - TServerSocket.h
  - TSocket.cxx
  - TSocket.h
- picture/
  - FitResult.cxx
  - FitResult.h
  - TF1.cxx
  - TF1.h
  - TF2.cxx
  - TF2.h
  - TF3.cxx
  - TF3.h
  - TFitResult.cxx
  - TFitResult.h
  - TFitResultPtr.cxx

    - TFitResultPtr.h
    - TGraph2D.cxx
    - TGraph2DErrors.cxx
    - TGraph2DErrors.h
    - TGraph2D.h
    - TGraph.cxx
    - TGraphErrors.cxx
    - TGraphErrors.h
    - TGraph.h
    - TGraphPolar.cxx
    - TGraphPolar.h
    - TH1.cxx
    - TH1.h
    - TH2.cxx
    - TH2.h
    - TH2Poly.cxx
    - TH2Poly.h
    - TH3.cxx
    - TH3.h
    - TMultiGraph.cxx
    - TMultiGraph.h
    - TPolyMarker3D.cxx
    - TPolyMarker3D.h
    - TPolyMarker.cxx
    - TPolyMarker.h
    - TProfile2D.cxx
    - TProfile2D.h
    - TProfile3D.cxx
    - TProfile3D.h
    - TProfile.cxx
    - TProfile.h
- roofit/
- Spectrum/
    - TSpectrum2.cxx
    - TSpectrum2Fit.cxx
    - TSpectrum2Fit.h

    - TSpectrum2.h
    - TSpectrum2Painter.cxx
    - TSpectrum2Painter.h
    - TSpectrum2Transform.cxx
    - TSpectrum2Transform.h
    - TSpectrum3.cxx
    - TSpectrum3.h
    - TSpectrum.cxx
    - TSpectrumFit.cxx
    - TSpectrumFit.h
    - TSpectrum.h
    - TSpectrumTransform.cxx
    - TSpectrumTransform.h
- thread/
    - TThread.cxx
    - TThreadFactory.cxx
    - TThreadFactory.h
    - TThread.h
    - TThreadImp.cxx
    - TThreadImp.h
    - TThreadPool.h
    - TThreadSlots.h
- time/
    - TBenchmark.cxx
    - TBenchmark.h
    - TStopwatch.cxx
    - TStopwatch.h
    - TTime.cxx
    - TTime.h
    - TTimer.cxx
    - TTimer.h
    - TTimeStamp.cxx
    - TTimeStamp.h
- tmva/
- TRandom/
    - TRandom1.cxx

    - TRandom1.h*
    - TRandom2.cxx
    - TRandom2.h*
    - TRandom3.cxx
    - TRandom3.h*
    - TRandom.cxx
    - TRandom.h*
- TSelector/
    - TSelectorCint.cxx
    - TSelectorCint.h
    - TSelector.cxx
    - TSelectorDraw.cxx
    - TSelectorDraw.h
    - TSelectorEntries.cxx
    - TSelectorEntries.h
    - TSelector.h
    - TSelectorList.cxx
    - TSelectorList.h
    - TSelectorScalar.cxx
    - TSelectorScalar.h
- system/
    - TDirectory.cxx
    - TDirectoryFile.cxx
    - TDirectoryFile.h*
    - TDirectory.h*
    - TNamed.cxx
    - TNamed.h*
    - TObject.cxx
    - TObject.h*
    - TROOT.cxx
    - TROOT.h*
    - TStyle.cxx
    - TStyle.h*
    - TSystem.cxx
    - TSystemDirectory.cxx
    - TSystemDirectory.h*

  - TSystemFile.cxx
  - TSystemFile.h*
  - TSystem.h*

- TEllipse.h
- TEllipse.cxx
- TString.cxx
- TString.h*
- TTask.cxx
- TTask.h*