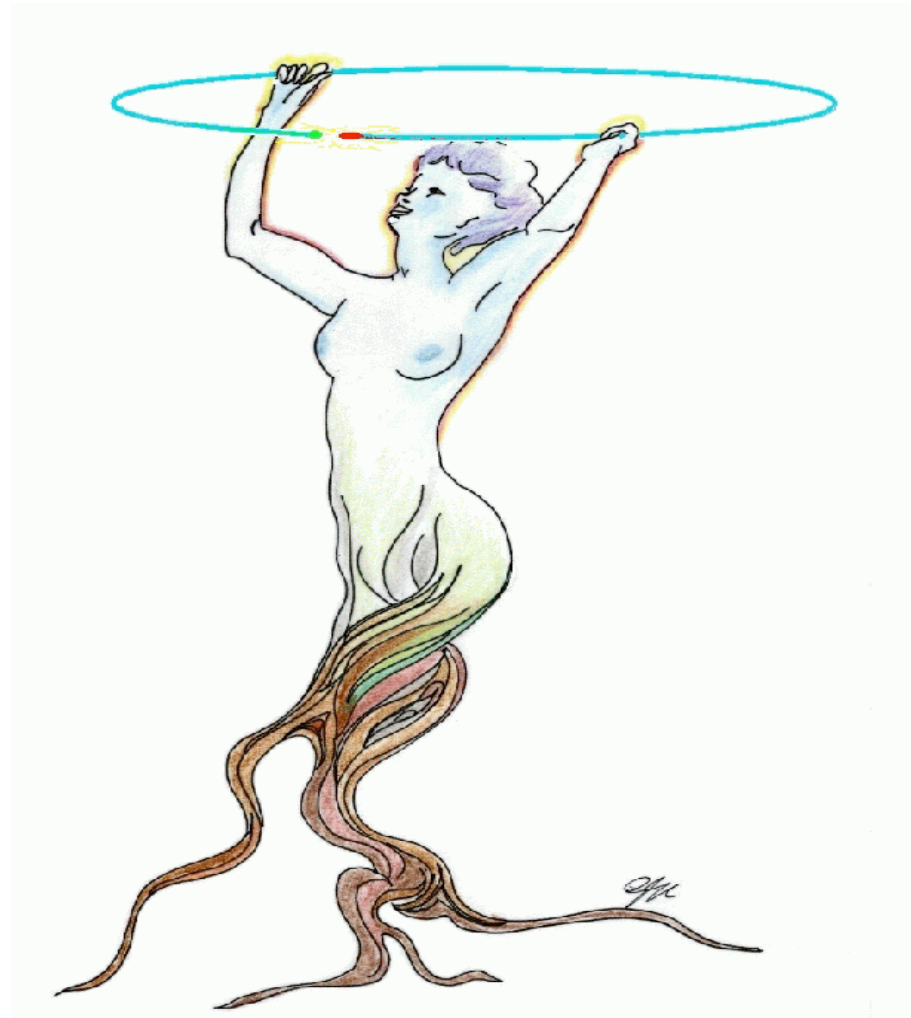


Formation ROOT pour débutants

Deuxième Jour
Programmation



Création et destruction d'objets

Les commandes "new" et "delete"

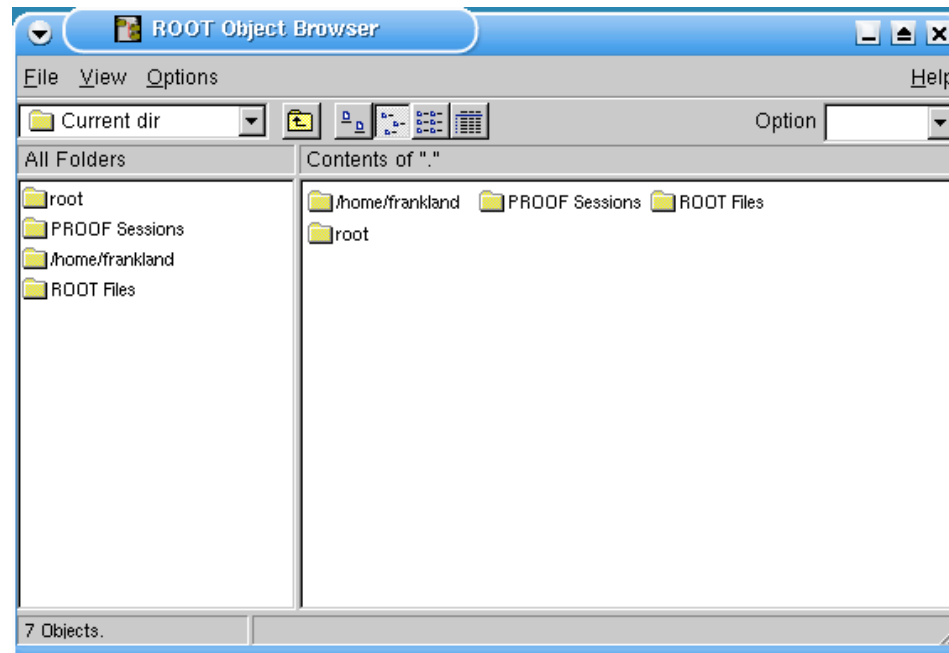
Les pointeurs d'objet

La commande "new"

- On a commencé la journée d'hier en tapant:

new TBrowser

ce qui a fait apparaître le "ROOT object browser".



La commande "new"

- On a commencé la journée d'hier en tapant:

new TBrowser

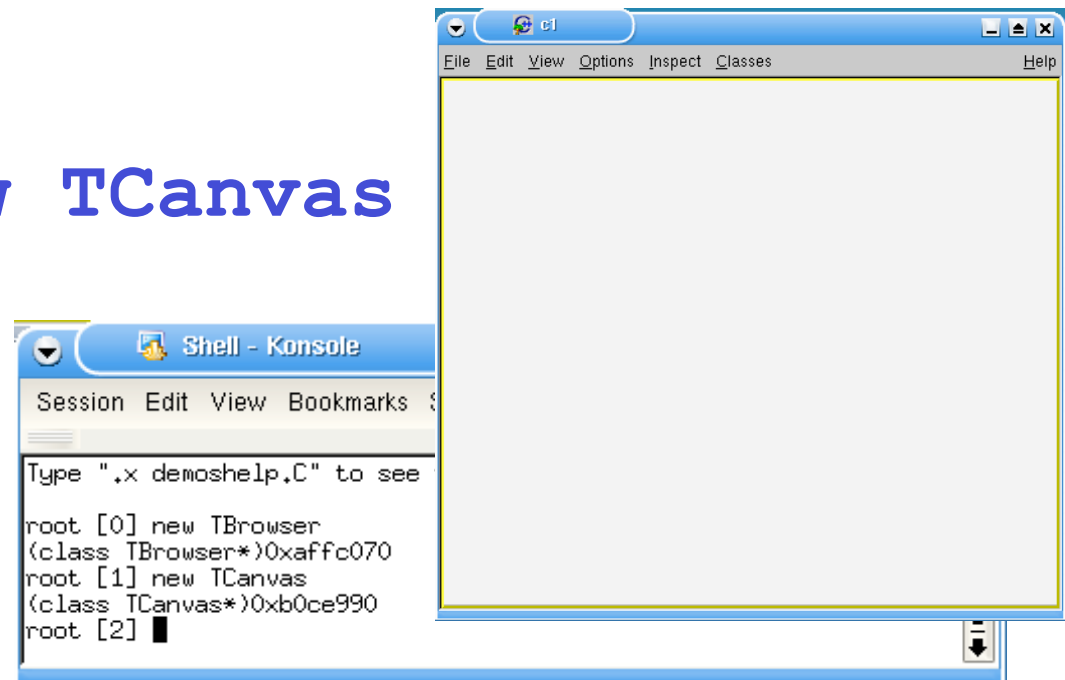
ce qui a fait apparaître le "ROOT object browser".

- Si l'on tape

new TCanvas

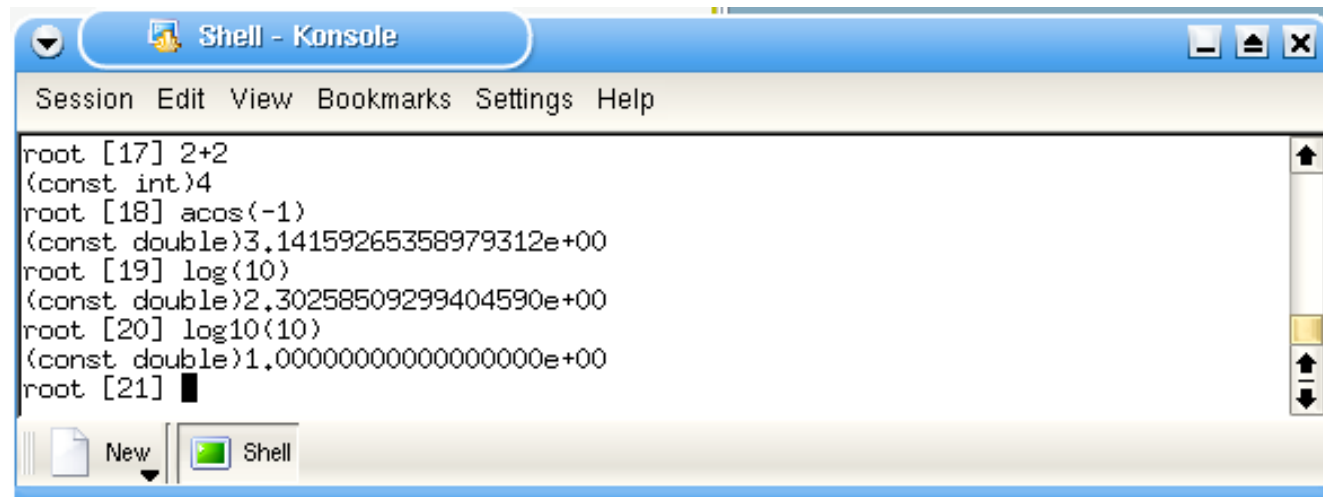
un canevas apparaît!

Mais qu'est-ce qui se passe
dans la fenêtre de
commandes ???



La fenêtre de commandes

- La fenêtre de commandes est un interpréteur de C++ !!
- L'interpréteur affiche la valeur de chaque fonction, expression ou commande que l'on tape - essayez e.g. $2+2$...



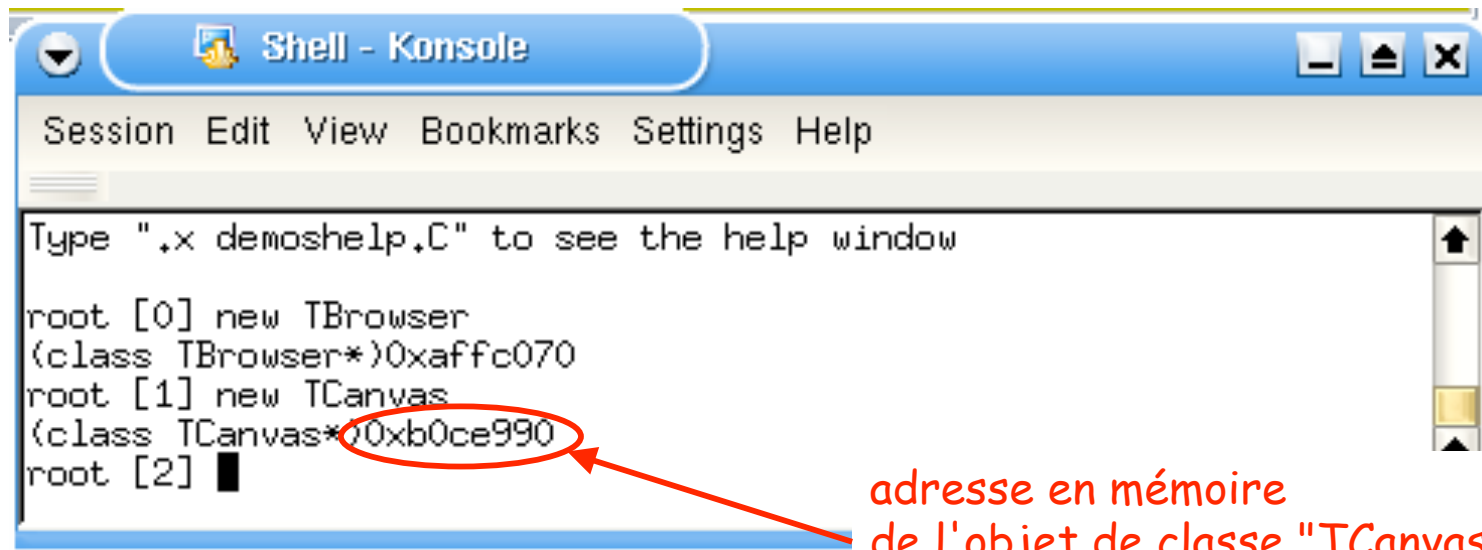
The screenshot shows a window titled "Shell - Konsole" with a menu bar containing "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The main area displays the following text:

```
root [17] 2+2
(const int)4
root [18] acos(-1)
(const double)3.14159265358979312e+00
root [19] log(10)
(const double)2.30258509299404590e+00
root [20] log10(10)
(const double)1.00000000000000000e+00
root [21] █
```

At the bottom, there is a toolbar with a "New" button and a "Shell" button.

La fenêtre de commandes

- Les valeurs affichées après une commande **new** sont le type (*classe*) et l'adresse en mémoire des objets créés



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Type ".x demoshelp.C" to see the help window
root [0] new TBrowser
(class TBrowser*)0xaffc070
root [1] new TCanvas
(class TCanvas*)0xb0ce990
root [2] █
```

adresse en mémoire
de l'objet de classe "TCanvas"
qui est affiché sur votre écran

Les pointeurs d'objet

- Pour utiliser l'objet, il faut mettre son adresse dans une variable spéciale, un *pointeur d'objet*

Déclaration d'un
pointeur d'objet : `TypeObjet* toto;`

Les pointeurs d'objet

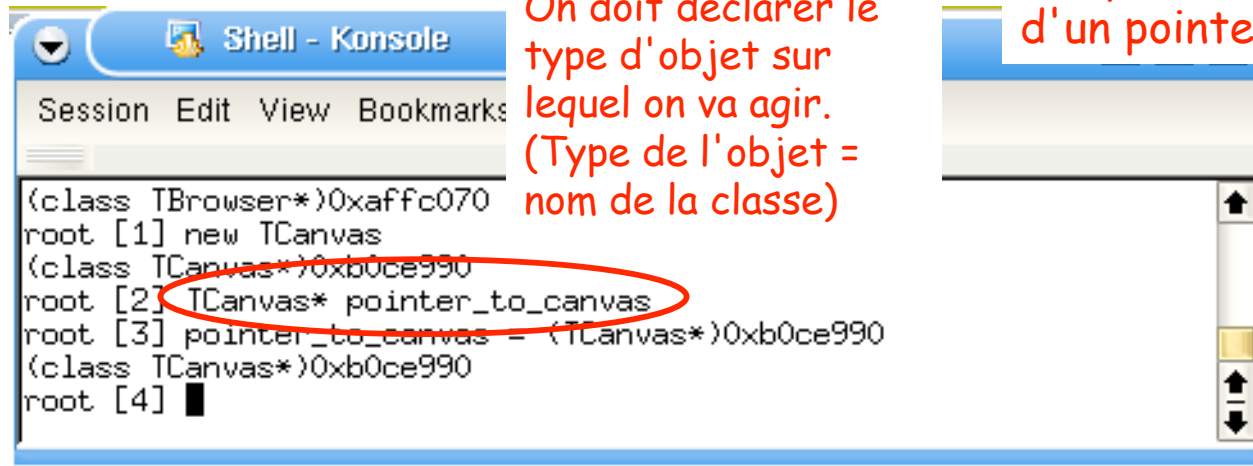
- Pour utiliser l'objet, il faut mettre son adresse dans une variable spéciale, un *pointeur d'objet*

Déclaration d'un
pointeur d'objet :

TypeObjet* toto;

On doit déclarer le
type d'objet sur
lequel on va agir.
(Type de l'objet =
nom de la classe)

C'est '*' qui nous
dit qu'il s'agit
d'un pointeur



```
Shell - Konsole
Session Edit View Bookmarks

(class TBrowser*)0xaffc070
root [1] new TCanvas
(class TCanvas*)0xb0ce990
root [2] TCanvas* pointer_to_canvas
root [3] pointer_to_canvas = (TCanvas*)0xb0ce990
(class TCanvas*)0xb0ce990
root [4] █
```


Les pointeurs d'objet

- La valeur stockée par le pointeur est réellement l'adresse de l'objet en mémoire

Initialisation d'un
pointeur d'objet :

```
toto = (TypeObjet*)adresse ;
```

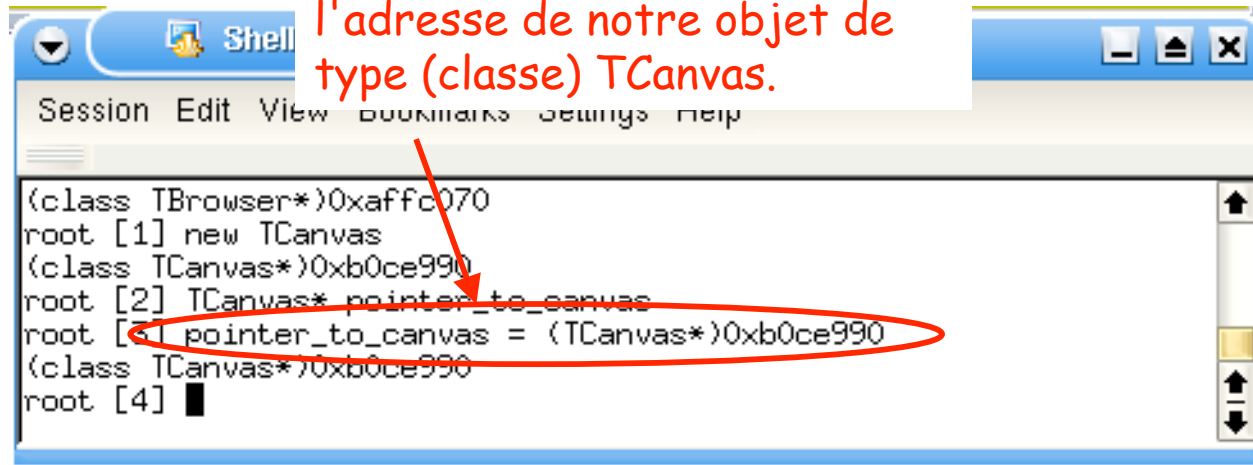
Les pointeurs d'objet

- La valeur stockée par le pointeur est réellement l'adresse de l'objet en mémoire

Initialisation d'un
pointeur d'objet :

`toto = (TypeObjet*)adresse ;`

On initialise le pointeur avec
l'adresse de notre objet de
type (classe) TCanvas.



```
Shell
Session Edit View bookmarks settings help

(class TBrowser*)0xaffc070
root [1] new TCanvas
(class TCanvas*)0xb0ce990
root [2] TCanvas* pointer_to_canvas
root [4] pointer_to_canvas = (TCanvas*)0xb0ce990
(class TCanvas*)0xb0ce990
root [4] █
```

Destruction d'objets

- La commande *delete* permet de libérer l'espace mémoire occupé par les objets
- On doit l'utiliser pour détruire les objets dont on n'a plus besoin, sinon on finira par remplir toute la mémoire!

Destruction d'un
objet:

```
delete toto;
```

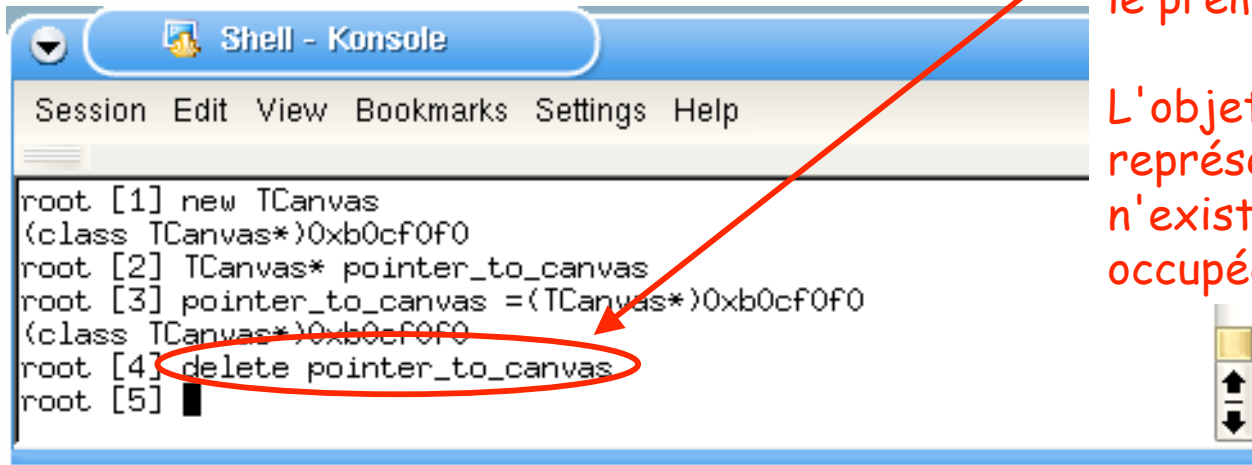
Destruction d'objets

- La commande *delete* permet de libérer l'espace mémoire occupé par les objets
- On doit l'utiliser pour détruire les objets dont on n'a plus besoin, sinon on finira par remplir toute la mémoire!

Destruction d'un objet:

`delete toto;`

L'exécution de cette commande fait disparaître le premier canevas!



```
root [1] new TCanvas
(class TCanvas*)0xb0cf0f0
root [2] TCanvas* pointer_to_canvas
root [3] pointer_to_canvas =(TCanvas*)0xb0cf0f0
(class TCanvas*)0xb0cf0f0
root [4] delete pointer_to_canvas
root [5] █
```

L'objet (et sa représentation graphique) n'existe plus, la mémoire occupée est libérée

Les constructeurs d'objets

- La plupart du temps on effectue la déclaration du pointeur, la création de l'objet et l'initialisation du pointeur avec l'adresse de l'objet en une seule ligne!

Création d'objet avec
initialisation de
pointeur:

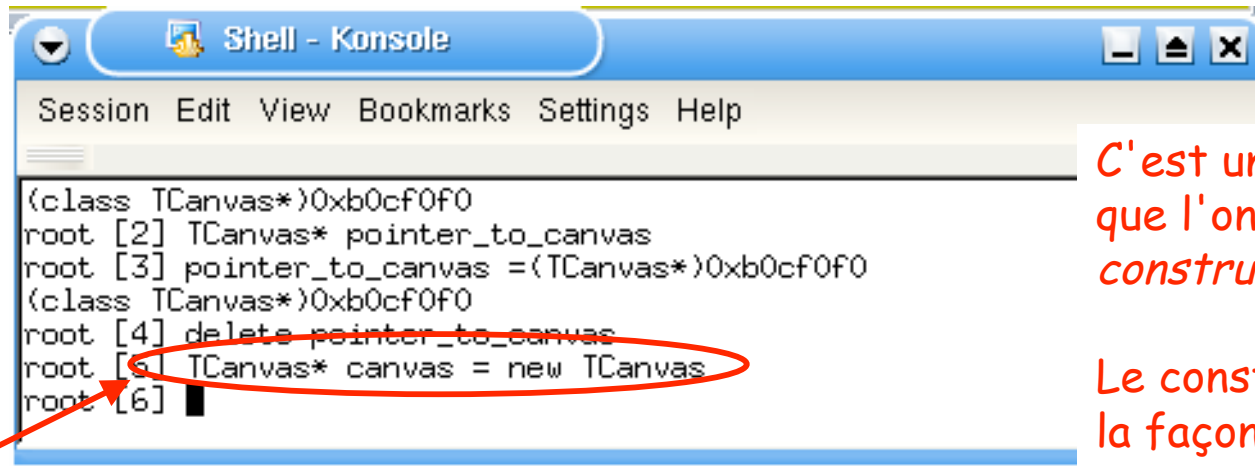
```
TypeObjet* toto = new TypeObjet;
```

Les constructeurs d'objets

- La plupart du temps on effectue la déclaration du pointeur, la création de l'objet et l'initialisation du pointeur avec l'adresse de l'objet en une seule ligne!

Création d'objet avec initialisation de pointeur:

```
TypeObjet* toto = new TypeObjet;
```



```
Shell - Konsole
Session Edit View Bookmarks Settings Help

(class TCanvas*)0xb0cf0f0
root [2] TCanvas* pointer_to_canvas
root [3] pointer_to_canvas =(TCanvas*)0xb0cf0f0
(class TCanvas*)0xb0cf0f0
root [4] delete pointer_to_canvas
root [5] TCanvas* canvas = new TCanvas
root [6]
```

C'est une fonction spéciale que l'on appelle un *constructeur*.

Le constructeur détermine la façon de créer les objets d'une classe.

*Un autre canevas apparaît!

Les constructeurs d'objets

- En règle générale, le constructeur peut prendre des arguments

Création d'objet avec
initialisation de
pointeur:

```
TypeObjet* toto = new TypeObjet(...);
```

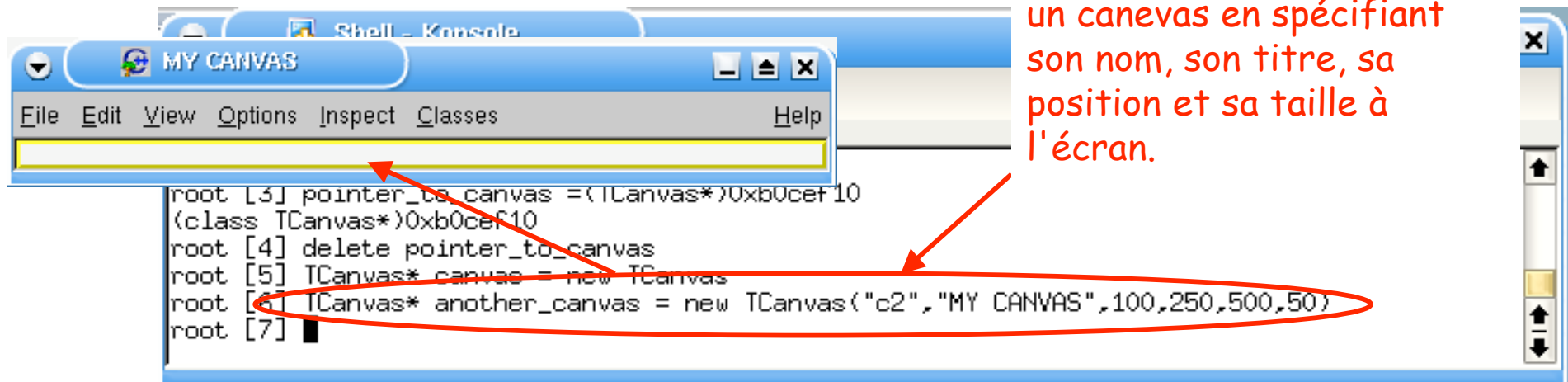
Les constructeurs d'objets

- En règle générale, le constructeur peut prendre des arguments

Création d'objet avec initialisation de pointeur:

`TypeObjet* toto = new TypeObjet(...);`

Par exemple, on peut créer un canevas en spécifiant son nom, son titre, sa position et sa taille à l'écran.



```
root [3] pointer_to_canvas = (TCanvas*)0xb0cef10
(class TCanvas*)0xb0cef10
root [4] delete pointer_to_canvas
root [5] TCanvas* canvas = new TCanvas
root [6] TCanvas* another_canvas = new TCanvas("c2", "MY CANVAS", 100, 250, 500, 50)
root [7]
```

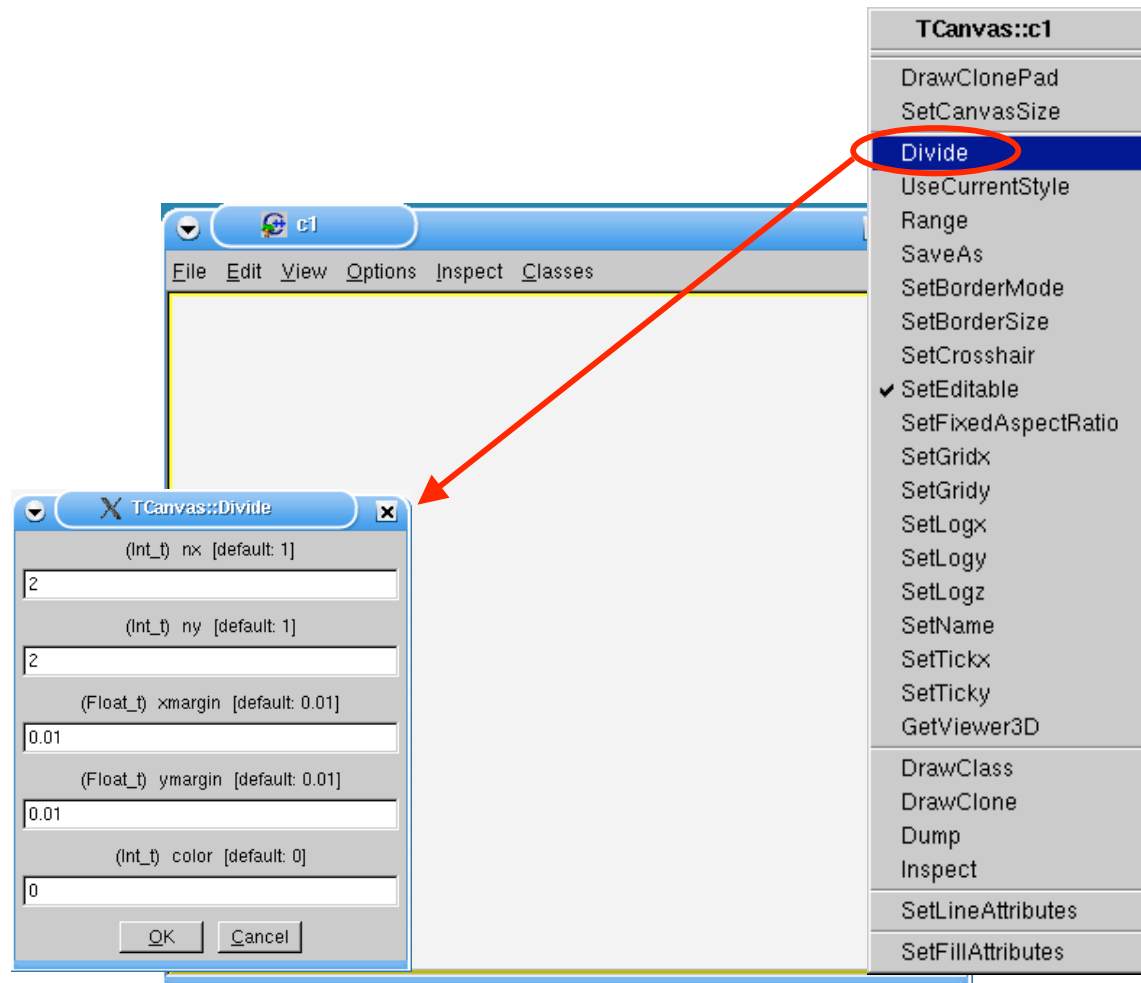
Dans l'exemple précédent, le canevas a par défaut le nom "c1" et le titre "c1"

Manipuler les objets

Les méthodes des classes

Agir sur les objets

- Graphiquement, on peut agir sur un objet à travers son menu contextuel:



Exemple d'hier,
quand on a divisé un
canevas en 4

Les méthodes

- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

toto-> Méthode(*arguments*);

Le pointeur d'objet avec l'opérateur '->' définit sur quel objet on va agir

Les méthodes

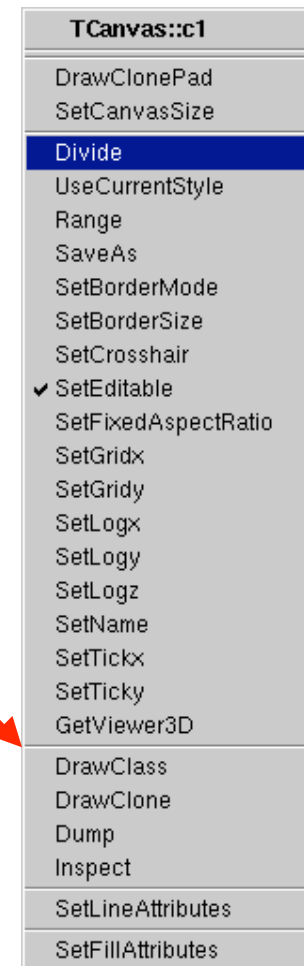
- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

toto-> **Méthode(arguments);**

Une des méthodes de la classe de l'objet. Par exemple, une des fonctions du menu contextuel.

Tous les objets de la même *classe* possèdent les mêmes *méthodes*.



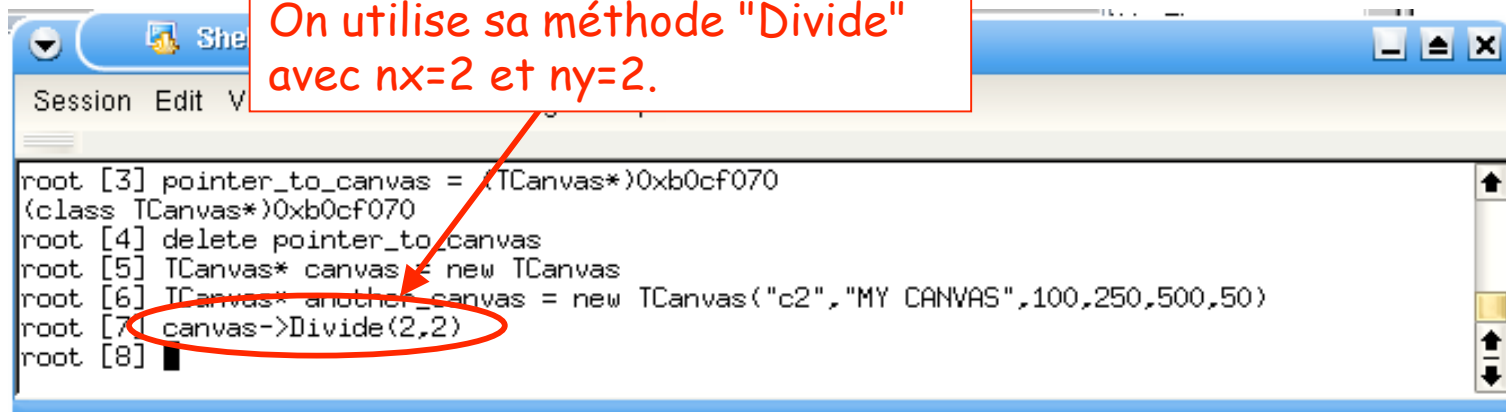
Les méthodes

- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

`canvas->Divide(2,2);`

On agit sur l'objet "c1" à travers son pointeur "canvas".
On utilise sa méthode "Divide" avec $nx=2$ et $ny=2$.



```
root [3] pointer_to_canvas = (TCanvas*)0xb0cf070
(class TCanvas*)0xb0cf070
root [4] delete pointer_to_canvas
root [5] TCanvas* canvas = new TCanvas
root [6] TCanvas* another_canvas = new TCanvas("c2", "MY CANVAS", 100, 250, 500, 50)
root [7] canvas->Divide(2,2)
root [8]
```

Les méthodes

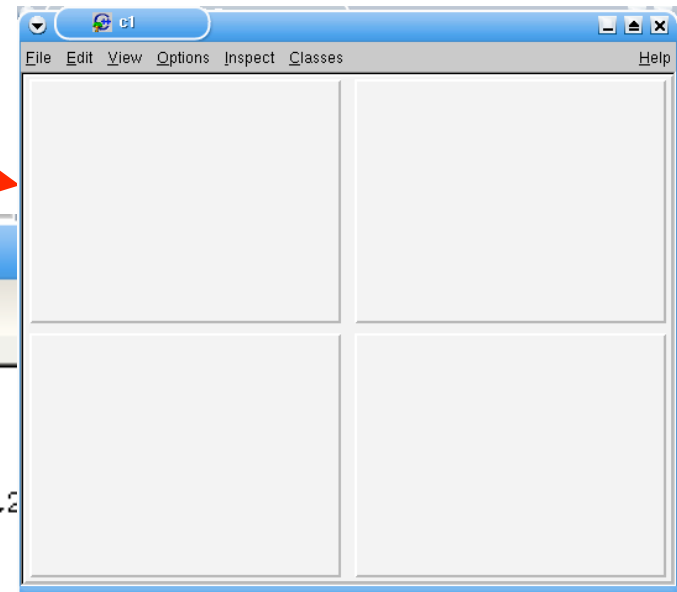
- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

`canvas->Divide(2,2);`

Le canevas "c1" se divise

```
Shell - Konsole
Session Edit View Bookmarks Settings Help
root [3] pointer_to_canvas = (TCanvas*)0xb0cf070
(class TCanvas*)0xb0cf070
root [4] delete pointer_to_canvas
root [5] TCanvas* canvas = new TCanvas
root [6] TCanvas* another_canvas = new TCanvas("c2", "MY CANVAS", 100, 2
root [7] canvas->Divide(2,2)
root [8] █
```



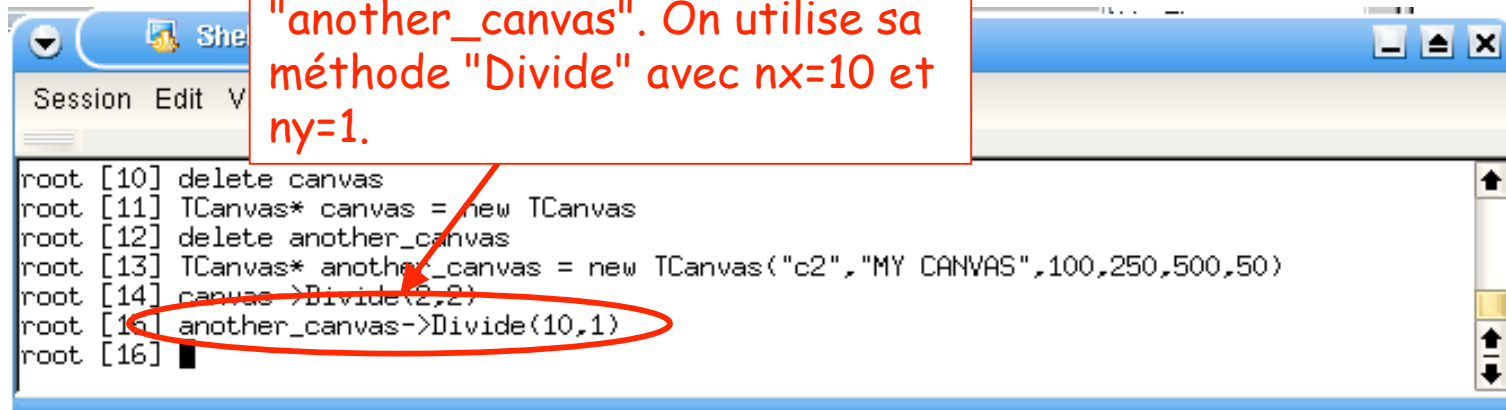
Les méthodes

- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

`another_canvas->Divide(10,1);`

On agit sur l'objet "c2" à travers son pointeur "another_canvas". On utilise sa méthode "Divide" avec nx=10 et ny=1.



```
root [10] delete canvas
root [11] TCanvas* canvas = new TCanvas
root [12] delete another_canvas
root [13] TCanvas* another_canvas = new TCanvas("c2","MY CANVAS",100,250,500,50)
root [14] canvas->Divide(2,2)
root [15] another_canvas->Divide(10,1)
root [16]
```

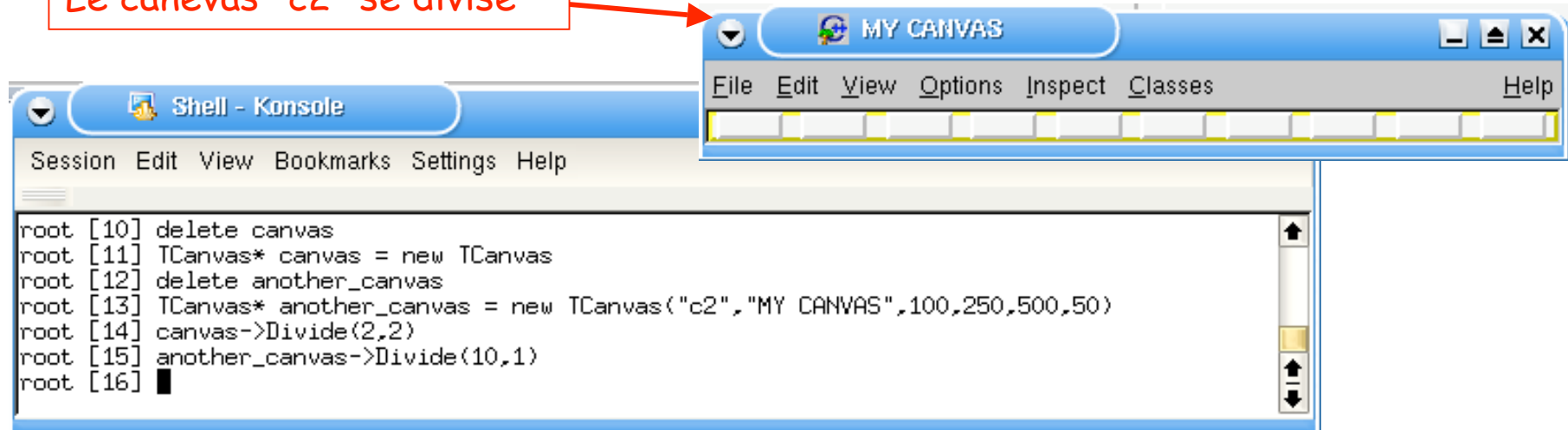
Les méthodes

- Avec un pointeur d'objet, on peut aussi interagir avec l'objet...

Agir sur un objet avec son pointeur:

`another_canvas->Divide(10,1);`

Le canevas "c2" se divise

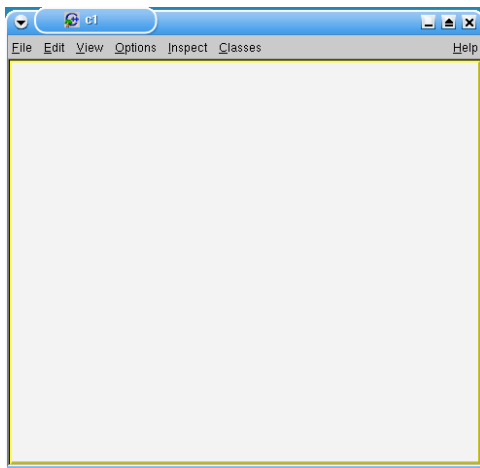
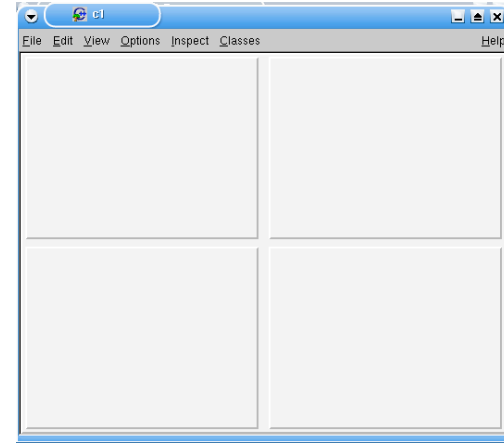


Les méthodes

- D'autres opérations sur les canevas:

`canvas->Clear();`

Effacer le contenu du
canevas, y compris les
divisions



Rendre le canevas
"actif", c-à-d c'est lui
qui aura un pourtour
jaune, et le prochain
histo s'affichera la-
dessus

`canvas->cd();`

Exemple avec un histogramme

- On peut créer un spectre à une dimension de la même façon que pour les canevas:

Création
d'un histo 1D

```
TH1F* histo = new TH1F("h1","My histo", 10, 0., 10.);
```

```
TypeObjet* toto = new TypeObjet(...);
```

REMARQUE: L'histo ne s'affiche pas
automatiquement!

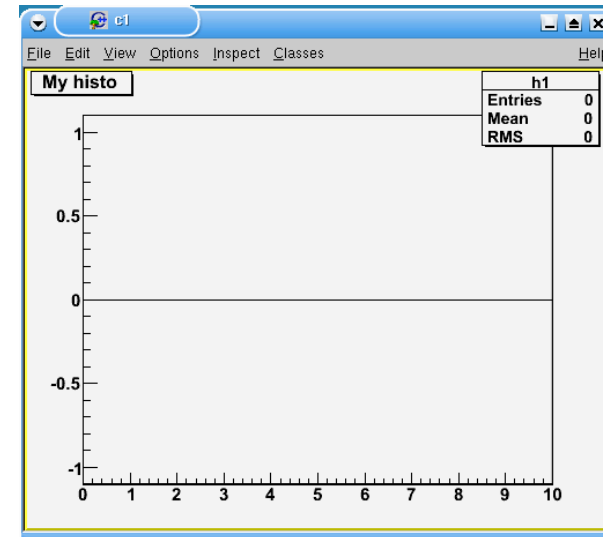
TH1F ? Histo à 1D de
nombres réels (Float)

Exemple avec un histogramme

- Afficher et remplir l'histogramme:

Afficher un spectre:

histo->Draw();



Remplir un spectre:

histo->Fill(3);

L'argument "3" correspond à une valeur de l'abscisse

Il ne s'est rien passé ?

Exemple avec un histogramme

- Mettre à jour l'affichage:

Signaler au
canevas qu'un
objet affiché a
été modifié:

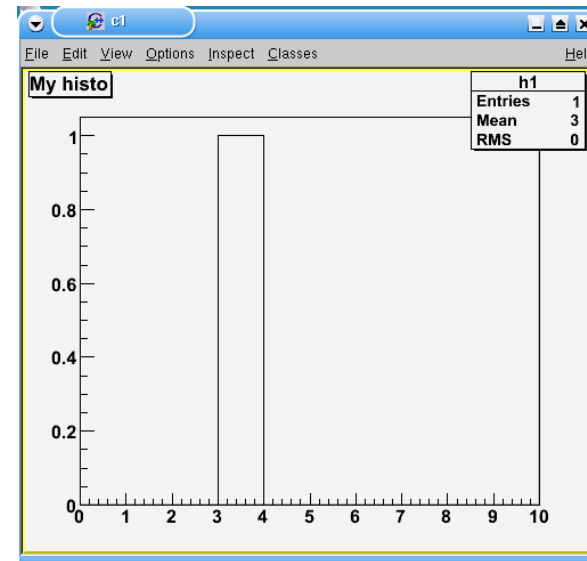
canvas->Modified();

A la ligne de commande, ça provoque
automatiquement la mise à jour du canevas.

On verra par la suite que la plupart du temps (e.g.
dans un programme) il faut aussi demander la mise
à jour!

Forcer la
mise à jour
du canevas:

canvas->Update();



Exemple avec un histogramme

- On commence à s'ennuyer?

On pourrait continuer comme ça jusqu'à ce que notre spectre soit rempli...

```
histo->Fill(1.43);  
histo->Fill(6.9);  
...  
histo->Fill(9, 2);  
canvas->Modified();
```

Remplissage
avec un poids



Mais on ferait mieux d'écrire une boucle...
...dans une fonction...
...en C++
...!!!

Création d'objets sans "new"

Il y a une autre façon de faire
Objets "temporaires" vs. objets
"permanents"

L'autre façon de faire...

- Il y a une autre façon de créer et de manipuler les objets...

Création d'objet sans
"new"

```
TypeObjet toto(...);
```

L'autre façon de faire...

- Il y a une autre façon de créer et de manipuler les objets...

Création d'objet sans
"new"

TypeObjet toto(...);

Si le constructeur prend des arguments, on les met ici

Création
d'objet avec
"new"

TypeObjet* toto_ptr = new TypeObjet(...);

L'autre façon de faire...

- Il y a une autre façon de créer et de manipuler les objets...

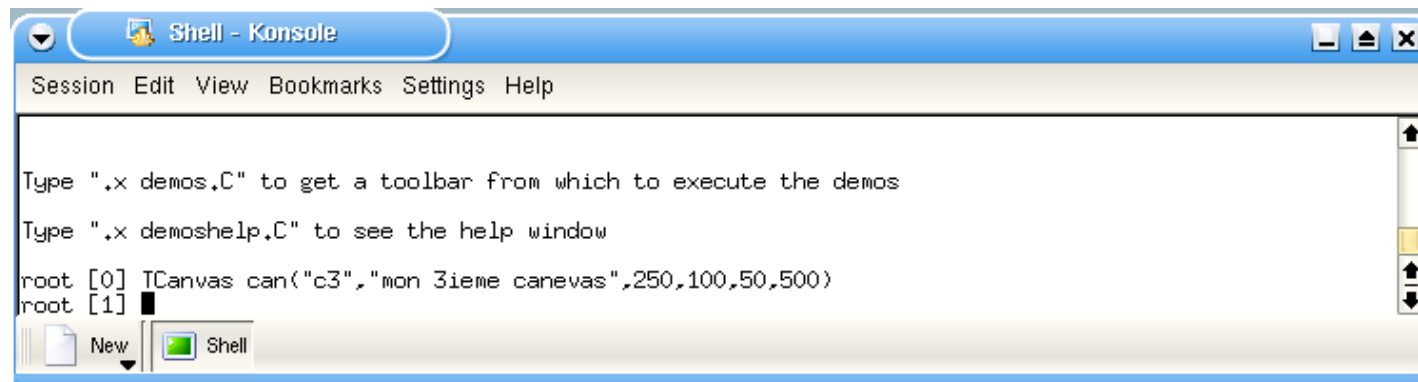
Création d'objet sans
"new"

```
TCanvas can("c3","titre",250,100,50,500);
```



Création
d'objet avec
"new"

```
TCanvas* can_ptr = new TCanvas("c3","titre", ...);
```

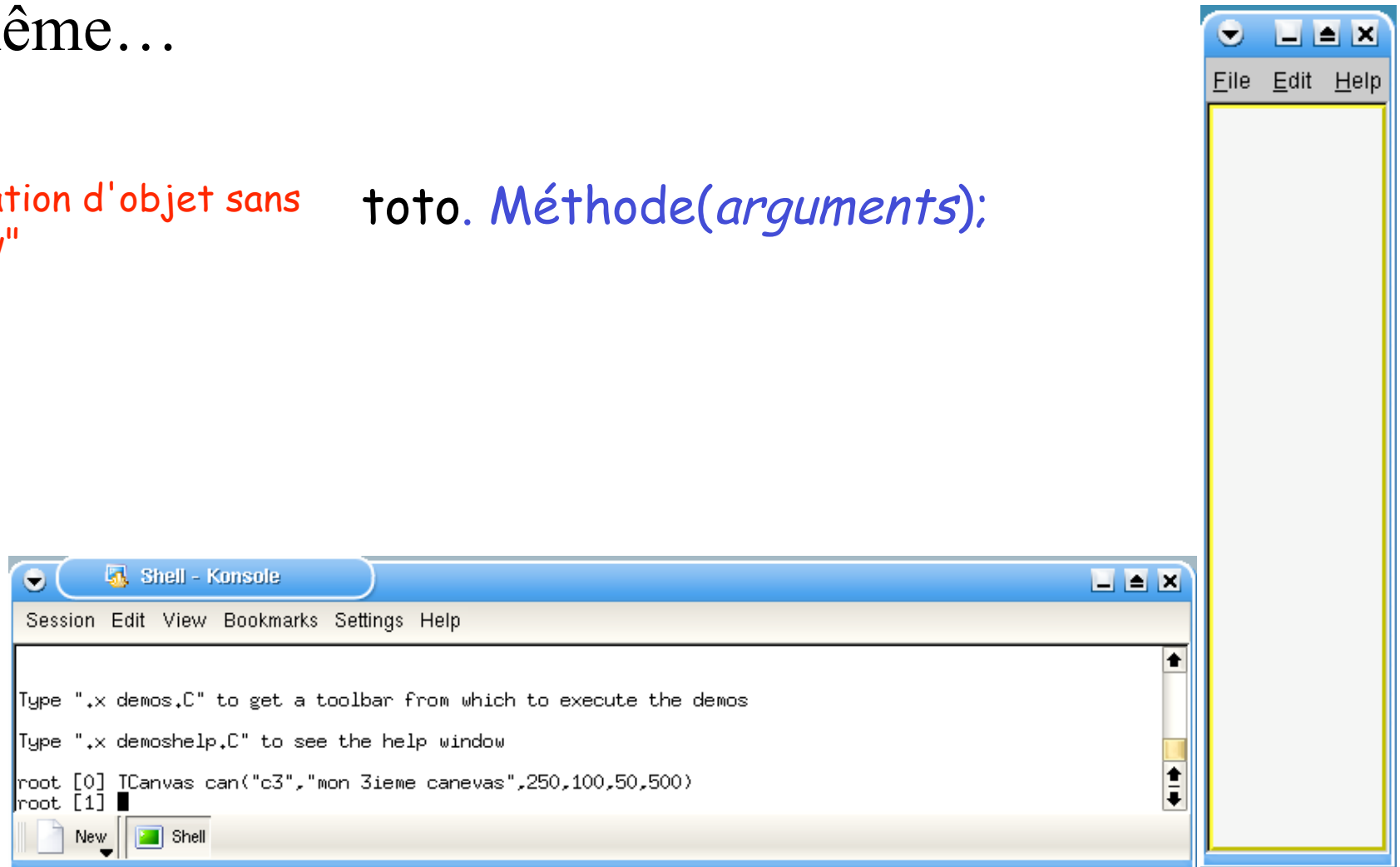


L'autre façon de faire...

- La façon d'agir sur l'objet n'est plus tout à fait la même...

Création d'objet sans
"new"

toto. *Méthode(arguments);*



L'autre façon de faire...

- La façon d'agir sur l'objet n'est plus tout à fait la même...

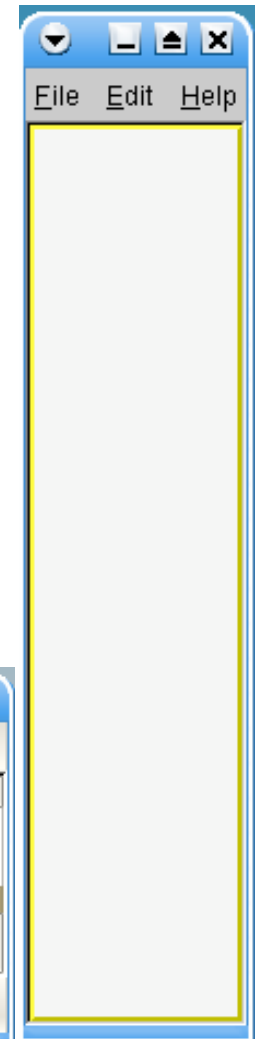
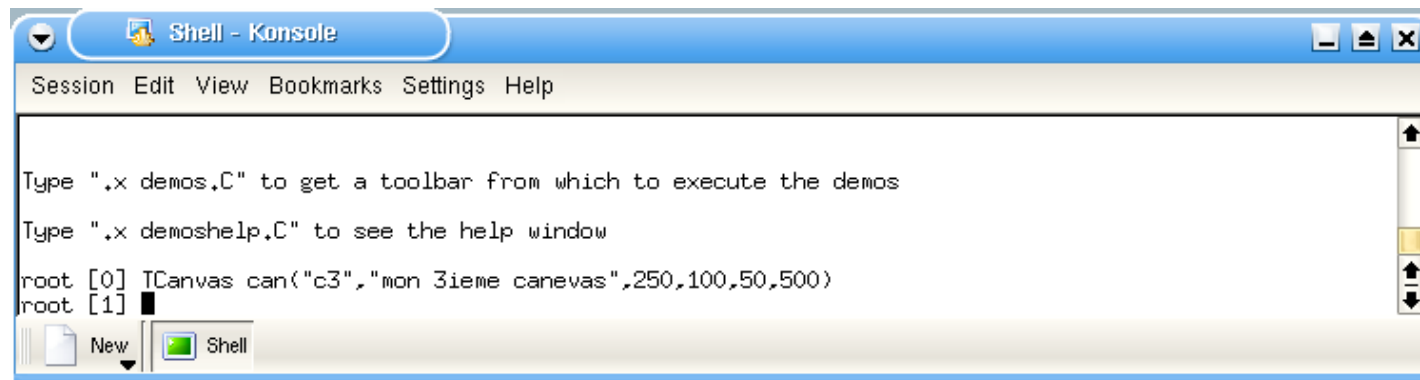
Création d'objet sans "new"

toto.Méthode(arguments);

Dans un cas on utilise un point, dans l'autre une flèche

Agir sur un objet avec son pointeur:

toto_ptr->Méthode(arguments);



L'autre façon de faire...

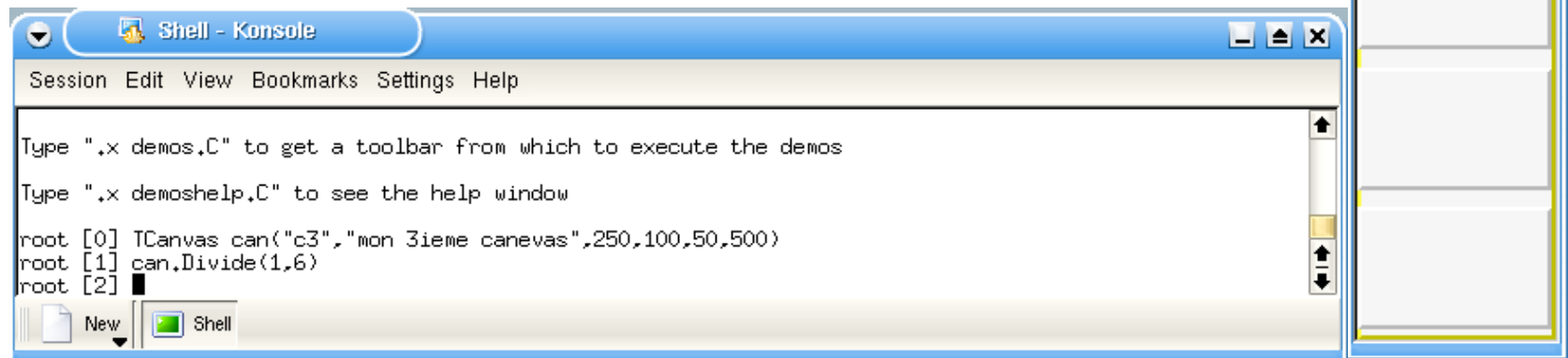
- La façon d'agir sur l'objet n'est plus tout à fait la même...

Création d'objet sans
"new"

`can.Divide(1,6);`

Agir sur un objet avec
son pointeur:

`can_ptr->Divide(1,6);`



L'autre façon de faire...

- On peut aussi obtenir l'adresse mémoire des objets créés de cette façon, et les manipuler en utilisant un *pointeur*

Initialiser un pointer
avec l'adresse d'un
objet existant

```
TypeObjet* toto_ptr = &toto;
```

L'autre façon de faire...

- On peut aussi obtenir l'adresse mémoire des objets créés de cette façon, et les manipuler en utilisant un *pointeur*

Initialiser un pointer avec l'adresse d'un objet existant

```
TypeObjet* toto_ptr = &toto;
```

Opérateur qui renvoie l'adresse mémoire de l'objet

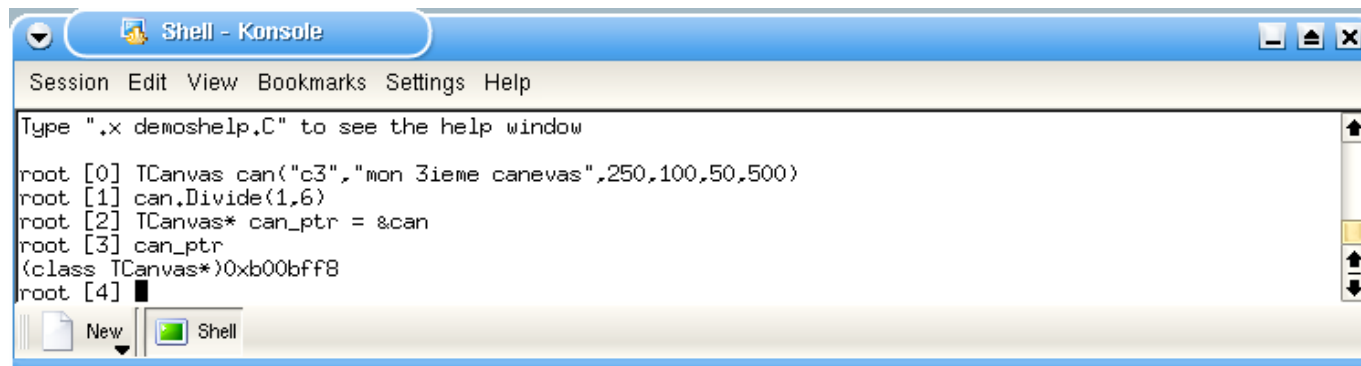
```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Type ".x demoshelp.C" to see the help window
root [0] TCanvas can("c3", "mon 3ieme canevas", 250, 100, 50, 500)
root [1] can.Divide(1,6)
root [2] TCanvas* can_ptr = &can
root [3] can_ptr
(class TCanvas*)0xb00bff8
root [4]
```

L'autre façon de faire...

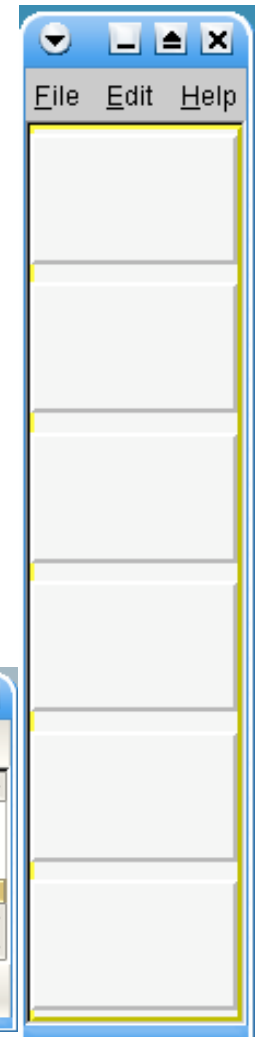
- L'utilisation du *pointeur* pour agir sur l'objet ensuite est identique aux cas précédents...

Agir sur l'objet à
travers son pointeur

toto_ptr ->Méthode(arguments);



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Type ".x demoshelp.C" to see the help window
root [0] TCanvas can("c3","mon 3ieme canevas",250,100,50,500)
root [1] can.Divide(1,6)
root [2] TCanvas* can_ptr = &can
root [3] can_ptr
(class TCanvas*)0xb00bff8
root [4] █
```

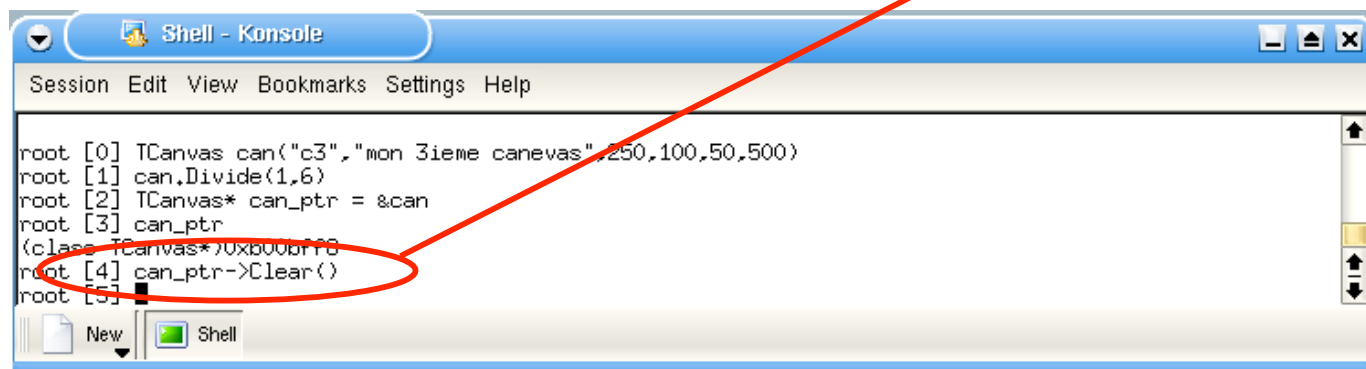


L'autre façon de faire...

- L'utilisation du *pointeur* pour agir sur l'objet ensuite est identique aux cas précédents...

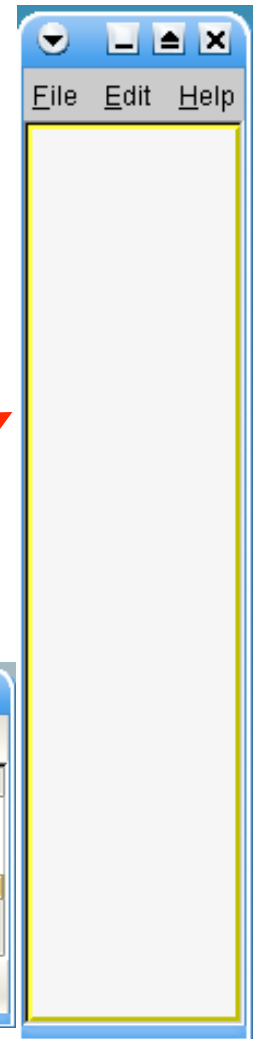
Agir sur l'objet à
travers son pointeur

```
can_ptr ->Clear();
```



```
Shell - Konsole
Session Edit View Bookmarks Settings Help

root [0] TCanvas can("c3","mon 3ieme canevas",250,100,50,500)
root [1] can.Divide(1,6)
root [2] TCanvas* can_ptr = &can
root [3] can_ptr
(class TCanvas*)0xb00000f0
root [4] can_ptr->Clear()
root [5]
```

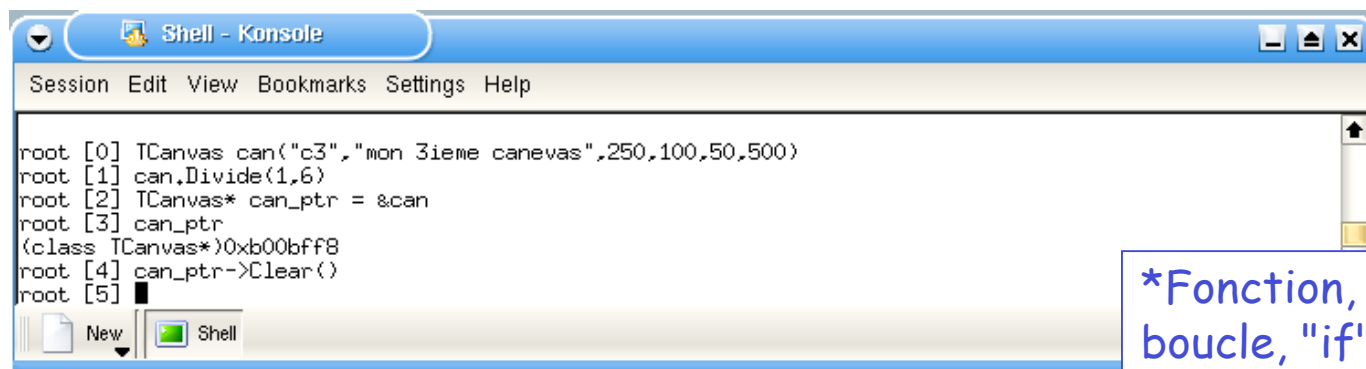


L'autre façon de faire...

- La différence ? On n'a pas besoin d'utiliser *delete* pour détruire l'objet quand on n'en veut plus...

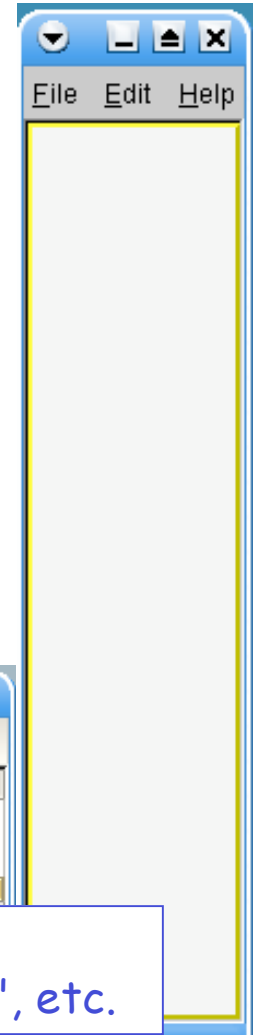
Les objets créés de cette façon sont détruits automatiquement à la fin du *bloc de code** dans lequel ils ont été créés.

Les objets créés avec "new" sont détruits à la demande de l'utilisateur avec "delete".



```
root [0] TCanvas can("c3","mon 3ieme canevas",250,100,50,500)
root [1] can.Divide(1,6)
root [2] TCanvas* can_ptr = &can
root [3] can_ptr
(class TCanvas*)0xb00bfff8
root [4] can_ptr->Clear()
root [5] █
```

*Fonction, boucle, "if", etc.



L'autre façon de faire...

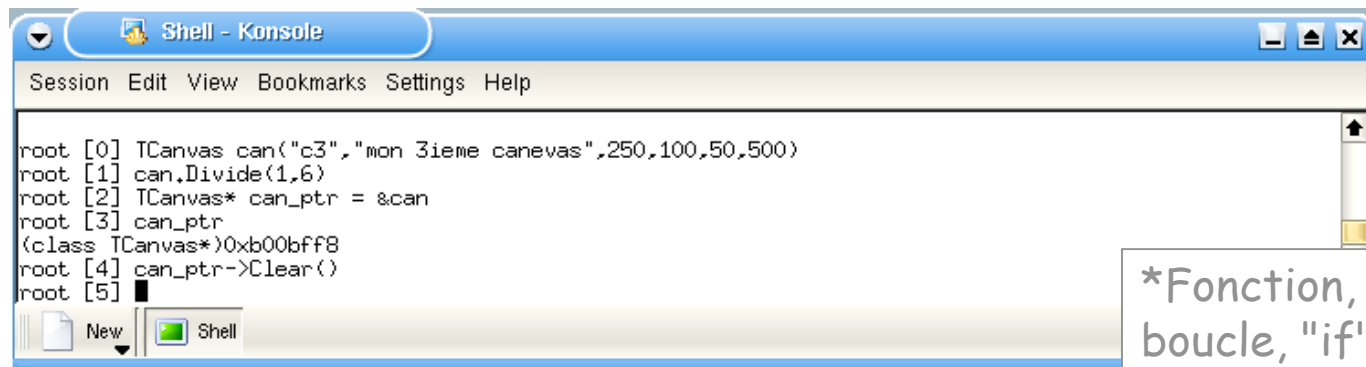
- La différence ? On n'a pas besoin d'utiliser *delete* pour détruire l'objet quand on n'en veut plus...

Objets
temporaires

Les objets créés de cette façon sont détruits automatiquement à la fin du *bloc de code** dans lequel ils ont été créés.

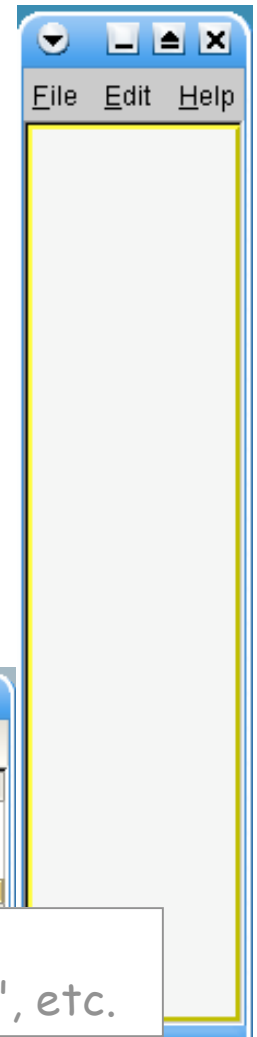
Objets
permanents

Les objets créés avec "new" sont détruits à la demande de l'utilisateur avec "delete".



```
root [0] TCanvas can("c3","mon 3ieme canevas",250,100,50,500)
root [1] can.Divide(1,6)
root [2] TCanvas* can_ptr = &can
root [3] can_ptr
(class TCanvas*)0xb00bfff8
root [4] can_ptr->Clear()
root [5] █
```

*Fonction,
boucle, "if", etc.



Trouver l'information sur les classes

Où est le mode d'emploi ?

Faut-il tout apprendre par cœur ?*

*NON!!!!

Trouver l'information

- Mais comment connaître toutes les façons d'interagir avec un objet ?
- Comment connaître toutes les méthodes d'une classe ?

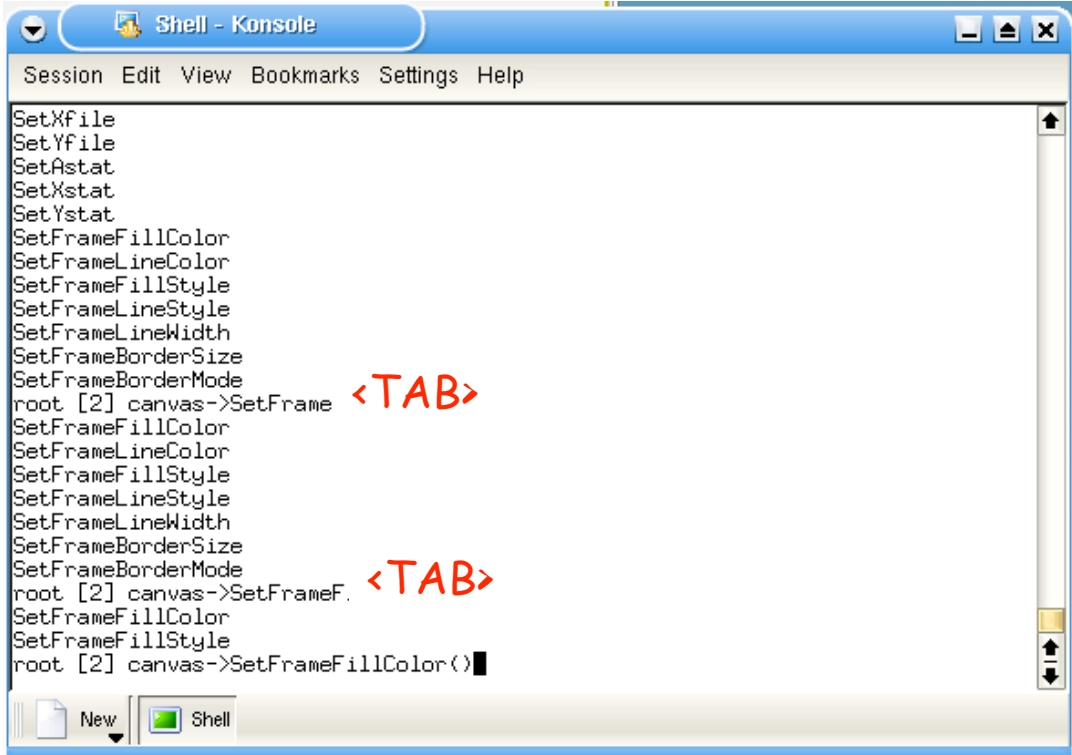
1. la complétion des commandes avec la touche <TAB>

Très efficace, permet de réduire au minimum les commandes à taper soi-même (c-à-d le nombre possible de fautes de frappe)...

ASTUCE:

la plupart des méthodes qui modifie un objet commencent par "Set..."

la plupart des méthodes qui renseignent sur un objet commencent par "Get..."



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
SetXfile
SetYfile
SetAstat
SetXstat
SetYstat
SetFrameFillColor
SetFrameLineColor
SetFrameFillStyle
SetFrameLineStyle
SetFrameLineWidth
SetFrameBorderSize
SetFrameBorderMode
root [2] canvas->SetFrame <TAB>
SetFrameFillColor
SetFrameLineColor
SetFrameFillStyle
SetFrameLineStyle
SetFrameLineWidth
SetFrameBorderSize
SetFrameBorderMode
root [2] canvas->SetFrameF. <TAB>
SetFrameFillColor
SetFrameFillStyle
root [2] canvas->SetFrameFillColor()
```

Trouver l'information

- La meilleur façon de se renseigner: consulter le site web <http://root.cern.ch>

2. Regarder le site web

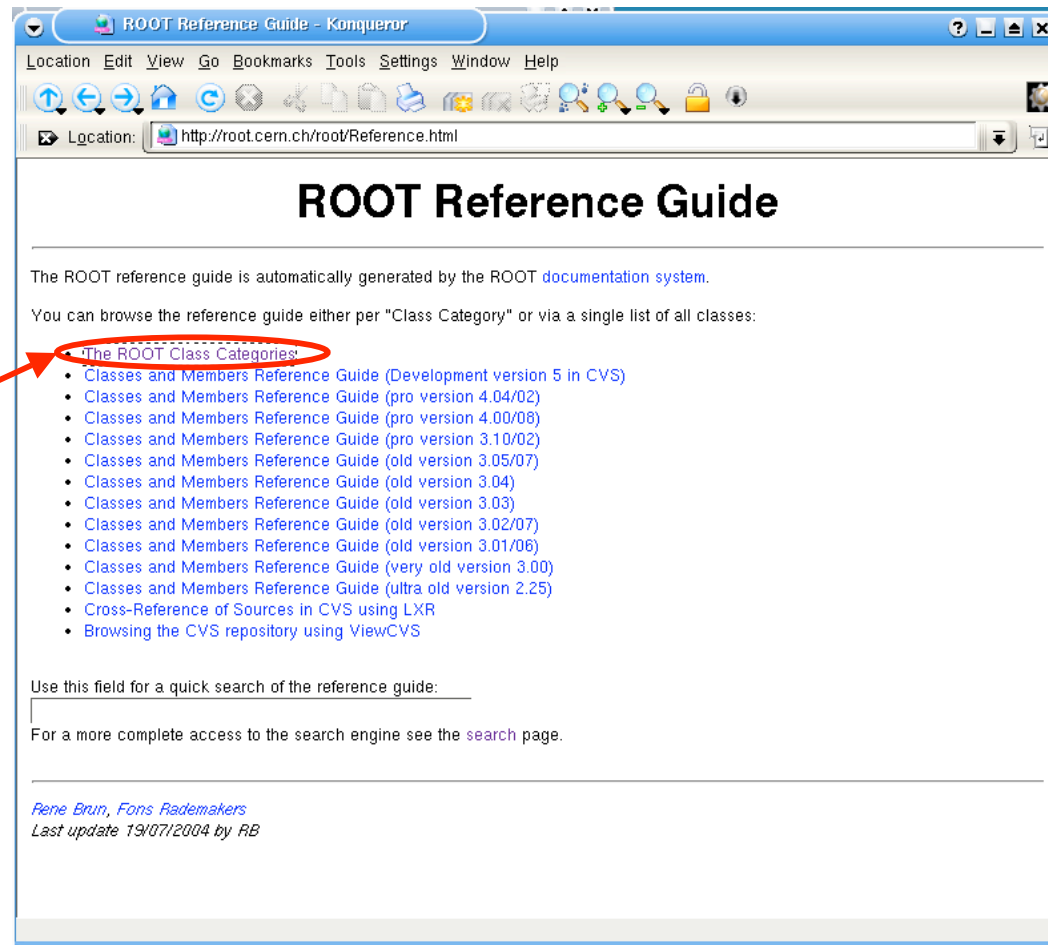
1. clique sur
"Reference Guide"

The screenshot shows the ROOT System Home Page in a Konqueror browser window. The browser title is "The ROOT System Home Page - Konqueror". The address bar shows "http://root.cern.ch/". The page content includes a navigation menu on the left with links such as "Roadmap", "Mission Statement", "Architecture", "Main Features", "CINT", "Coding Conventions", "Benchmarking", "Picture Gallery", "Publication List", "The ROOT Team", "License", "Register as User", "Download Binaries", "Install from Source", "CVS", "ViewCVS", "LXR", "User's Guide", "Reference Guide", "Tutorials", "HOWTO's", "RootTalk Forum", "RootTalk Digest", "Example Applications", "BaBar Tutorials", "FNAL Tutorials", "MINOS Tutorials", "PROOF", "Report a Bug", "Feedback", "Search", "News", and "Quick search:". The "Reference Guide" link is circled in red. The main content area features the ROOT logo and the text "An Object-Oriented Data Analysis Framework" next to a figure of a woman holding a ring. The news section at the bottom right contains the following text: "Production Version 4.04/02 **NEW** 04/05/2005", "The Production release of ROOT 4.04/02 is now available.", "See THIS IMPORTANT ANNOUNCEMENT.", "See also the announcement of the ROOT2005 Workshop.", "Tar files for the source, documentation and binaries are available at:", "Version 4.04/02 Release Notes", "<http://root.cern.ch/root/Version404.html>", and "The CVS tag for this version is **v4-04-02**."

Consultation du site web

- La meilleur façon de se renseigner: consulter le site web <http://root.cern.ch>

2. liste des classes
par catégories
(histos, matrices,
géométrie, etc.)



Consultation du site web

- La meilleur façon de se renseigner: consulter le site web <http://root.cern.ch>

2. liste complète
des classes pour
chaque version de
ROOT

ROOT Reference Guide - Konqueror

Location Edit View Go Bookmarks Tools Settings Window Help

Location: <http://root.cern.ch/root/Reference.html>

ROOT Reference Guide

The ROOT reference guide is automatically generated by the ROOT [documentation system](#).

You can browse the reference guide either per "Class Category" or via a single list of all classes:

- [The ROOT Class Categories](#)
- [Classes and Members Reference Guide \(Development version 5 in CVS\)](#)
- [Classes and Members Reference Guide \(pro version 4.04/02\)](#)
- [Classes and Members Reference Guide \(pro version 4.00/08\)](#)
- [Classes and Members Reference Guide \(pro version 3.10/02\)](#)
- [Classes and Members Reference Guide \(old version 3.05/07\)](#)
- [Classes and Members Reference Guide \(old version 3.04\)](#)
- [Classes and Members Reference Guide \(old version 3.03\)](#)
- [Classes and Members Reference Guide \(old version 3.02/07\)](#)
- [Classes and Members Reference Guide \(old version 3.01/06\)](#)
- [Classes and Members Reference Guide \(very old version 3.00\)](#)
- [Classes and Members Reference Guide \(ultra old version 2.25\)](#)
- [Cross-Reference of Sources in CVS using LXR](#)
- [Browsing the CVS repository using ViewCVS](#)

Use this field for a quick search of the reference guide: _____

For a more complete access to the search engine see the [search](#) page.

Rene Brun, Fons Rademakers
Last update 19/07/2004 by RB

Consultation du site web

- La meilleure façon de se renseigner: consulter le site web <http://root.cern.ch>

2. recherche
sur mots clés

ROOT Reference Guide - Konqueror

Location Edit View Go Bookmarks Tools Settings Window Help

Location: <http://root.cern.ch/root/Reference.html>

ROOT Reference Guide

The ROOT reference guide is automatically generated by the ROOT [documentation system](#).

You can browse the reference guide either per "Class Category" or via a single list of all classes:

- [The ROOT Class Categories](#)
- [Classes and Members Reference Guide \(Development version 5 in CVS\)](#)
- [Classes and Members Reference Guide \(pro version 4.04/02\)](#)
- [Classes and Members Reference Guide \(pro version 4.00/08\)](#)
- [Classes and Members Reference Guide \(pro version 3.10/02\)](#)
- [Classes and Members Reference Guide \(old version 3.05/07\)](#)
- [Classes and Members Reference Guide \(old version 3.04\)](#)
- [Classes and Members Reference Guide \(old version 3.03\)](#)
- [Classes and Members Reference Guide \(old version 3.02/07\)](#)
- [Classes and Members Reference Guide \(old version 3.01/06\)](#)
- [Classes and Members Reference Guide \(very old version 3.00\)](#)
- [Classes and Members Reference Guide \(ultra old version 2.25\)](#)
- [Cross-Reference of Sources in CVS using LXR](#)
- [Browsing the CVS repository using ViewCVS](#)

Use this field for a quick search of the reference guide:

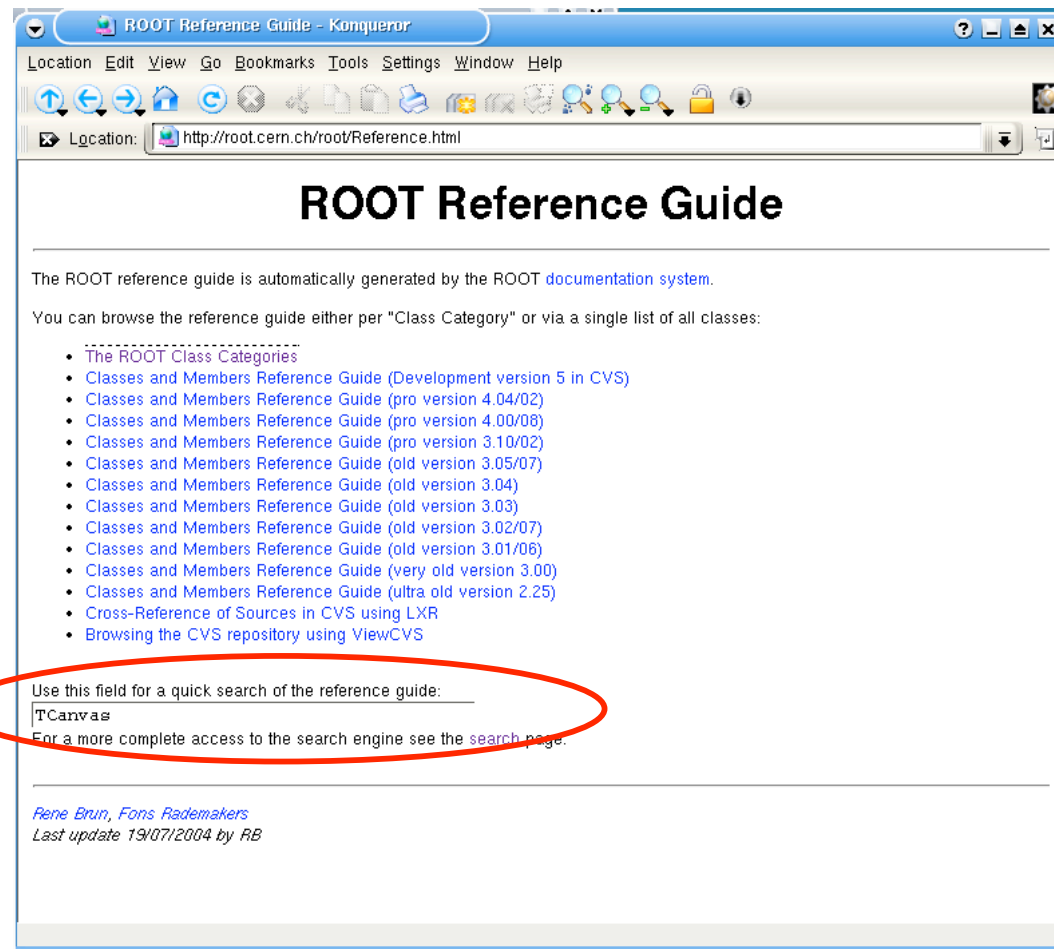
For a more complete access to the search engine see the [search page](#).

Rene Brun, Fons Rademakers
Last update 19/07/2004 by RB

Consultation du site web

- La meilleur façon de se renseigner: consulter le site web <http://root.cern.ch>

rentrez
"TCanvas"
(puis <ENTER>)



ROOT Reference Guide - Konqueror

Location Edit View Go Bookmarks Tools Settings Window Help

Location: <http://root.cern.ch/root/Reference.html>

ROOT Reference Guide

The ROOT reference guide is automatically generated by the ROOT [documentation system](#).

You can browse the reference guide either per "Class Category" or via a single list of all classes:

- [The ROOT Class Categories](#)
- [Classes and Members Reference Guide \(Development version 5 in CVS\)](#)
- [Classes and Members Reference Guide \(pro version 4.04/02\)](#)
- [Classes and Members Reference Guide \(pro version 4.00/08\)](#)
- [Classes and Members Reference Guide \(pro version 3.10/02\)](#)
- [Classes and Members Reference Guide \(old version 3.05/07\)](#)
- [Classes and Members Reference Guide \(old version 3.04\)](#)
- [Classes and Members Reference Guide \(old version 3.03\)](#)
- [Classes and Members Reference Guide \(old version 3.02/07\)](#)
- [Classes and Members Reference Guide \(old version 3.01/06\)](#)
- [Classes and Members Reference Guide \(very old version 3.00\)](#)
- [Classes and Members Reference Guide \(ultra old version 2.25\)](#)
- [Cross-Reference of Sources in CVS using LXR](#)
- [Browsing the CVS repository using ViewCVS](#)

Use this field for a quick search of the reference guide:

For a more complete access to the search engine see the [search page](#).

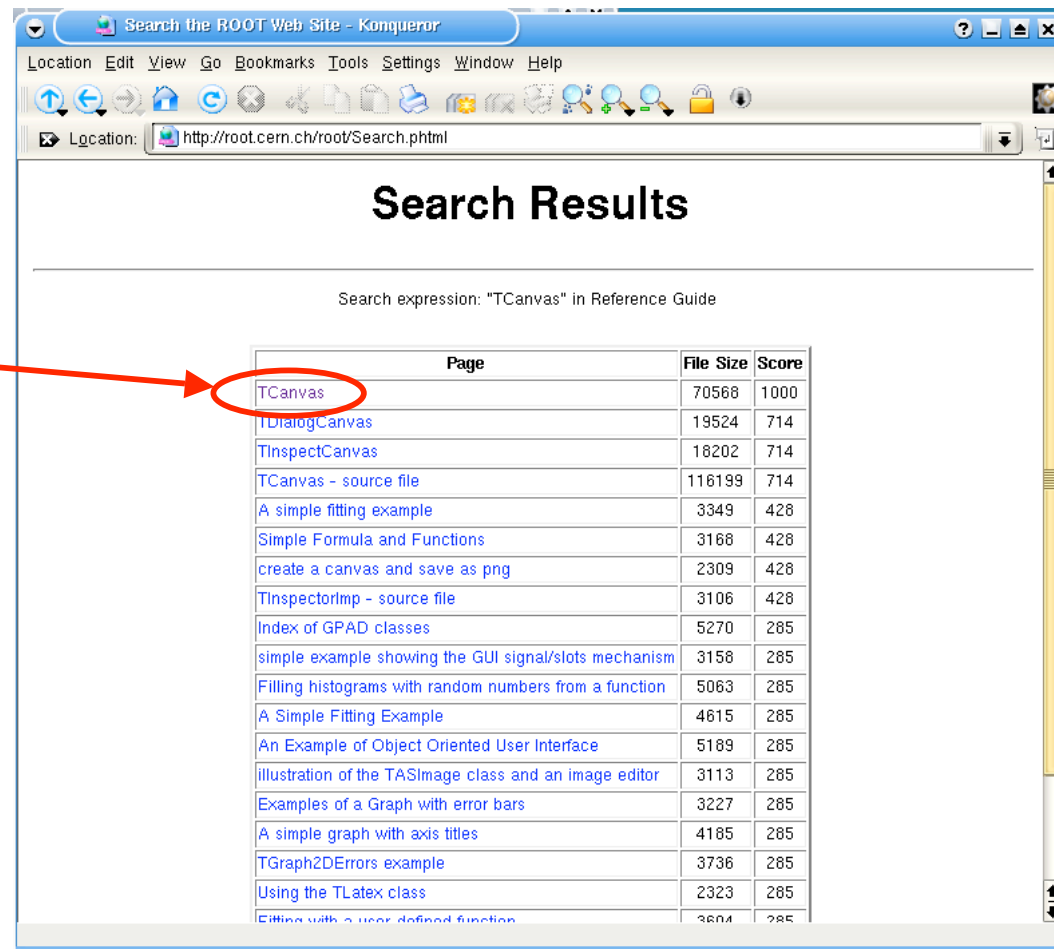
Rene Brun, Fons Rademakers
Last update 19/07/2004 by RB

Consultation du site web

- La meilleure façon de se renseigner: consulter le site web <http://root.cern.ch>

la recherche par nom de classe marche le mieux...

...mais ne négligez pas les autres résultats qui peuvent être très intéressants!!



Page	File Size	Score
TCanvas	70568	1000
TDialogCanvas	19524	714
TInspectCanvas	18202	714
TCanvas - source file	116199	714
A simple fitting example	3349	428
Simple Formula and Functions	3168	428
create a canvas and save as png	2309	428
TInspectorImp - source file	3106	428
Index of GPAD classes	5270	285
simple example showing the GUI signal/slots mechanism	3158	285
Filling histograms with random numbers from a function	5063	285
A Simple Fitting Example	4615	285
An Example of Object Oriented User Interface	5189	285
illustration of the TASImage class and an image editor	3113	285
Examples of a Graph with error bars	3227	285
A simple graph with axis titles	4185	285
TGraph2DErrors example	3736	285
Using the TLatex class	2323	285
Fitting with a user defined function	3604	285

Présentation d'une page de description d'une classe

- On y trouve toutes les informations nécessaires... à condition de savoir les trouver

Le nom de la classe et de son parent le plus proche (classe "mère")

class **TCanvas** : public **TPad**

Inheritance Chart:

```
graph TD
    TCanvas --> TPad
    TPad --> TVirtualPad
    TVirtualPad --> TAttFill
    TAttFill --> TAttLine
    TAttLine --> TObject
    TAttLine --> TAttMarkerCanvas
    TAttLine --> TAttTextCanvas
    TAttLine --> TDialogCanvas
    TAttLine --> TDrawPanelHist
    TAttLine --> TFitPanel
    TFitPanel --> TFitPanelGraph
    TFitPanel --> TInspectCanvas
    TAttLine --> TQObject
```

private:

```
TCanvas(const TCanvas& canvas)
void Build()
virtual void CopyPixmaps()
void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
TCanvas& operator=(const TCanvas& rhs)
void DrawAutoExec()
```

Présentation d'une page de description d'une classe

- On y trouve toutes les informations nécessaires... à condition de savoir les trouver

Généalogie complète de la classe

TCanvas

library: libGpad
#include "TCanvas.h"

[class description - source file - inheritance tree \(.pdf\)](#)

class TCanvas : public TPad

Inheritance Chart:

```
graph TD
    TCanvas --> TPad
    TCanvas --> TAttFillCanvas
    TCanvas --> TAttLineCanvas
    TCanvas --> TAttMarkerCanvas
    TCanvas --> TAttTextCanvas
    TCanvas --> TDialogCanvas
    TCanvas --> TDrawPanelHist
    TCanvas --> TFitPanel
    TCanvas --> TFitPanelGraph
    TCanvas --> TInspectCanvas
    TPad --> TVirtualPad
    TVirtualPad --> TAttFill
    TAttFill --> TAttLine
    TAttLine --> TAttText
    TAttText --> TAttMarker
    TDialogCanvas --> TDrawPanelHist
    TFitPanel --> TFitPanelGraph
```

private:

```
TCanvas(const TCanvas& canvas)
void Build()
virtual void CopyPixmaps()
void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
TCanvas& operator=(const TCanvas& rhs)
void DrawAutoExec()
```

Présentation d'une page de description d'une classe

- On y trouve toutes les informations nécessaires... à condition de savoir les trouver

Généalogie complète de la classe

parents et grand-parents à gauche

library: libGpad
#include "TCanvas.h"

TCanvas

[class description - source file - inheritance tree \(.pdf\)](#)

```
class TCanvas : public TPad
```

Inheritance Chart:

```
graph TD
    TObject --> TAttLine
    TObject --> TAttFill
    TObject --> TAttPad
    TObject --> TQObject
    TAttLine --> TAttFill
    TAttLine --> TAttPad
    TAttFill --> TVirtualPad
    TVirtualPad --> TPad
    TPad --> TCanvas
    TCanvas --> TAttFillCanvas
    TCanvas --> TAttLineCanvas
    TCanvas --> TAttMarkerCanvas
    TCanvas --> TAttTextCanvas
    TCanvas --> TDialogCanvas
    TCanvas --> TDrawPanelHist
    TCanvas --> TFitPanel
    TCanvas --> TInspectCanvas
    TFitPanel --> TFitPanelGraph
```

```
private:
    TCanvas(const TCanvas& canvas)
    void Build()
    virtual void CopyPixmaps()
    void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
    TCanvas& operator=(const TCanvas& rhs)
    void DrawAutoExec()
```

Présentation d'une page de description d'une classe

- On y trouve toutes les informations nécessaires... à condition de savoir les trouver

Généalogie complète de la classe

parents et grands-parents à gauche

enfants et petits-enfants à droite

Location: http://root.cern.ch/cgi-bin/print_hit_bold.pl/root/html/TCanvas.html?TCanvas#first_hit

TCanvas

library: libGpad
#include "TCanvas.h"

[class description](#) - [source file](#) - [inheritance tree \(.pdf\)](#)

```
class TCanvas : public TPad
```

Inheritance Chart:

```
graph TD
    TCanvas --> TPad
    TPad --> TVirtualPad
    TVirtualPad --> TAttFill
    TCanvas --> TDialogCanvas
    TCanvas --> TAttFillCanvas
    TCanvas --> TAttLineCanvas
    TCanvas --> TAttMarkerCanvas
    TCanvas --> TAttTextCanvas
    TCanvas --> TDrawPanelHist
    TCanvas --> TFitPanel
    TCanvas --> TInspectCanvas
```

private:

```
TCanvas(const TCanvas& canvas)
void Build()
virtual void CopyPixmaps()
void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
TCanvas& operator=(const TCanvas& rhs)
void DrawAutoExec()
```

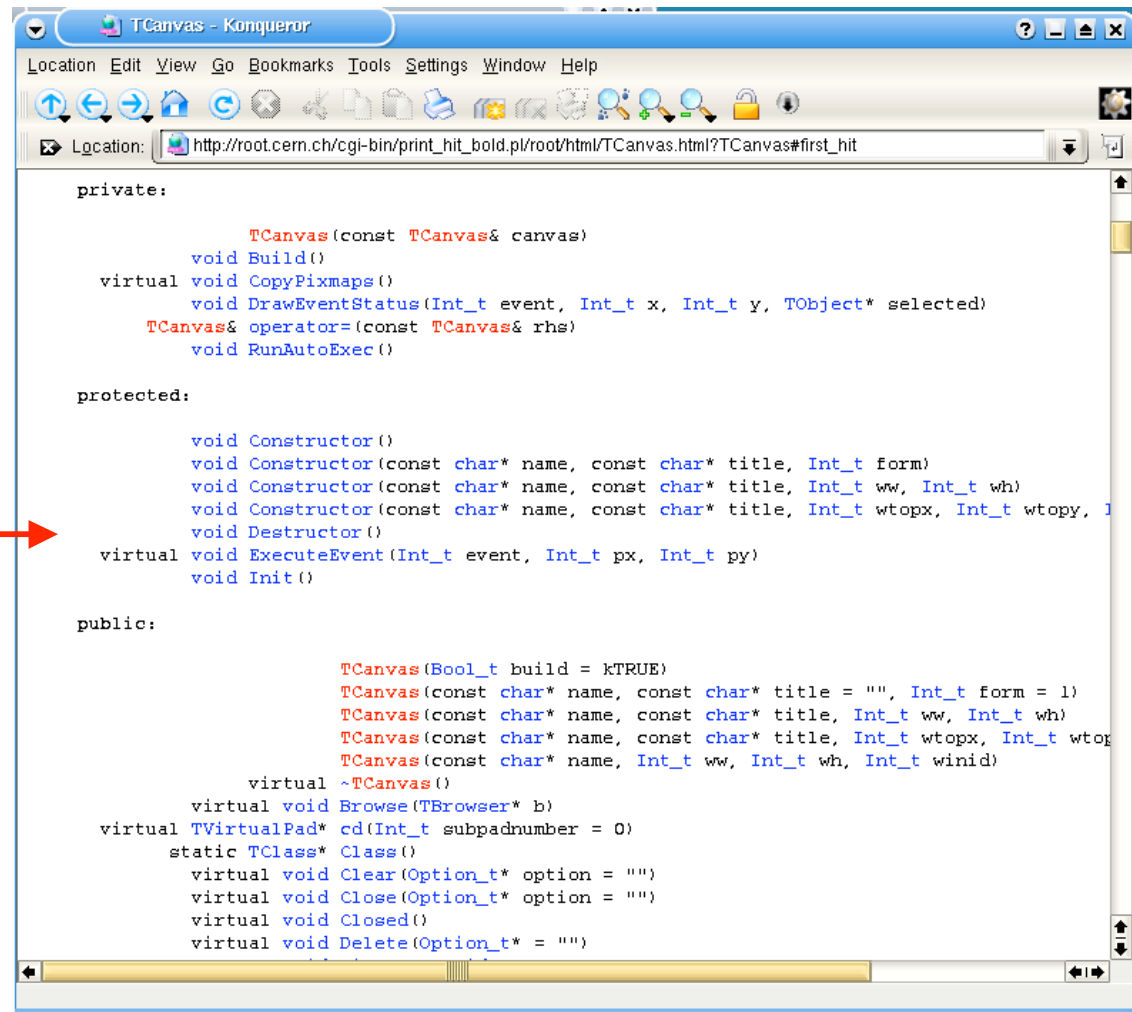
La liste des méthodes d'une classe

- On y trouve toutes les informations nécessaires... à condition de savoir les trouver

(un peu plus bas)

la liste complète des méthodes de la classe

chaque nom est un lien vers une documentation complète de la méthode



The screenshot shows a web browser window titled "TCanvas - Konqueror". The address bar contains the URL "http://root.cern.ch/cgi-bin/print_hit_bold.pl/root/html/TCanvas.html?TCanvas#first_hit". The main content area displays the C++ method list for the TCanvas class, organized into sections: private, protected, and public. The methods listed include constructors, virtual methods like CopyPixmaps, DrawEventStatus, ExecuteEvent, and public methods like Browse, cd, Class, Clear, Close, Closed, and Delete. A red arrow points from the text "la liste complète des méthodes de la classe" to the "protected:" section of the code.

```
private:
    TCanvas(const TCanvas& canvas)
    void Build()
    virtual void CopyPixmaps()
    void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
    TCanvas& operator=(const TCanvas& rhs)
    void RunAutoExec()

protected:
    void Constructor()
    void Constructor(const char* name, const char* title, Int_t form)
    void Constructor(const char* name, const char* title, Int_t ww, Int_t wh)
    void Constructor(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t wright, Int_t wbottom)
    void Destructor()
    virtual void ExecuteEvent(Int_t event, Int_t px, Int_t py)
    void Init()

public:
    TCanvas(Bool_t build = kTRUE)
    TCanvas(const char* name, const char* title = "", Int_t form = 1)
    TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
    TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t wright, Int_t wbottom)
    TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)

    virtual ~TCanvas()
    virtual void Browse(TBrowser* b)
    virtual TVirtualPad* cd(Int_t subpadnumber = 0)
    static TClass* Class()
    virtual void Clear(Option_t* option = "")
    virtual void Close(Option_t* option = "")
    virtual void Closed()
    virtual void Delete(Option_t* option = "")
```

La liste des méthodes d'une classe

- Il existe trois types de méthodes: "private", "protected", "public"

on ne peut utiliser ni les méthodes "private" ni "protected"...

... donc on ne regardera que les méthodes "public"

```
TCanvas - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://root.cern.ch/cgi-bin/print_hit_bold.pl/root/html/TCanvas.html?TCanvas#first_hit

private:
    TCanvas(const TCanvas& canvas)
    void Build()
    virtual void CopyPixmaps()
    void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)
    TCanvas& operator=(const TCanvas& rhs)
    void RunAutoExec()

protected:
    void Constructor()
    void Constructor(const char* name, const char* title, Int_t form)
    void Constructor(const char* name, const char* title, Int_t ww, Int_t wh)
    void Constructor(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t wright, Int_t wright)
    void Destructor()
    virtual void ExecuteEvent(Int_t event, Int_t px, Int_t py)
    void Init()

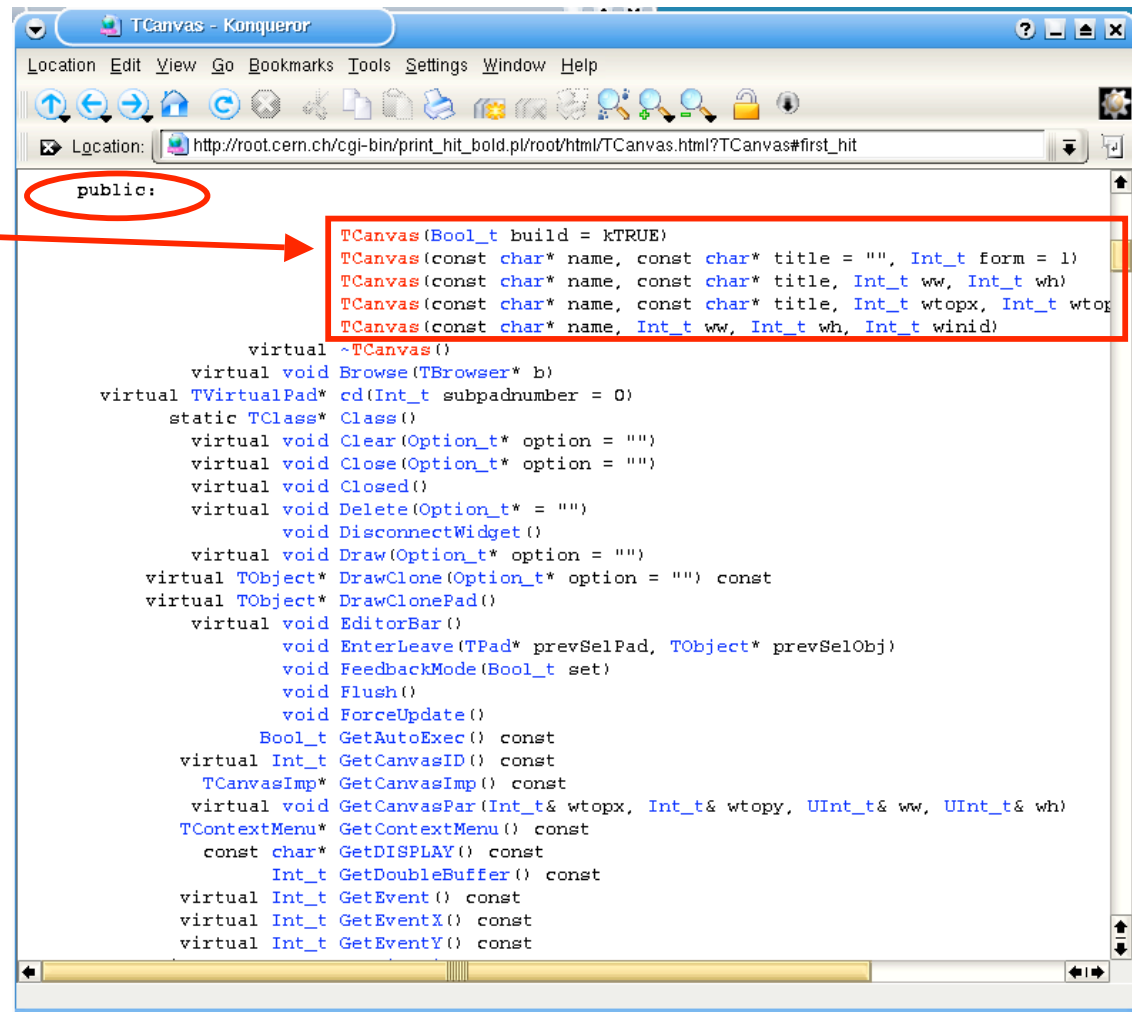
public:
    TCanvas(Bool_t build = kTRUE)
    TCanvas(const char* name, const char* title = "", Int_t form = 1)
    TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
    TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t wright, Int_t wright)
    TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)

    virtual ~TCanvas()
    virtual void Browse(TBrowser* b)
    virtual TVirtualPad* cd(Int_t subpadnumber = 0)
    static TClass* Class()
    virtual void Clear(Option_t* option = "")
    virtual void Close(Option_t* option = "")
    virtual void Closed()
    virtual void Delete(Option_t* option = "")
```


La liste des méthodes d'une classe

- La liste des méthodes "public" est toujours organisée de la même façon:

D'abord, on a les *constructeurs* de la classe (méthodes avec le même nom que la classe)



```
public:
TCanvas(Bool_t build = kTRUE)
TCanvas(const char* name, const char* title = "", Int_t form = 1)
TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t ww, Int_t wh)
TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)

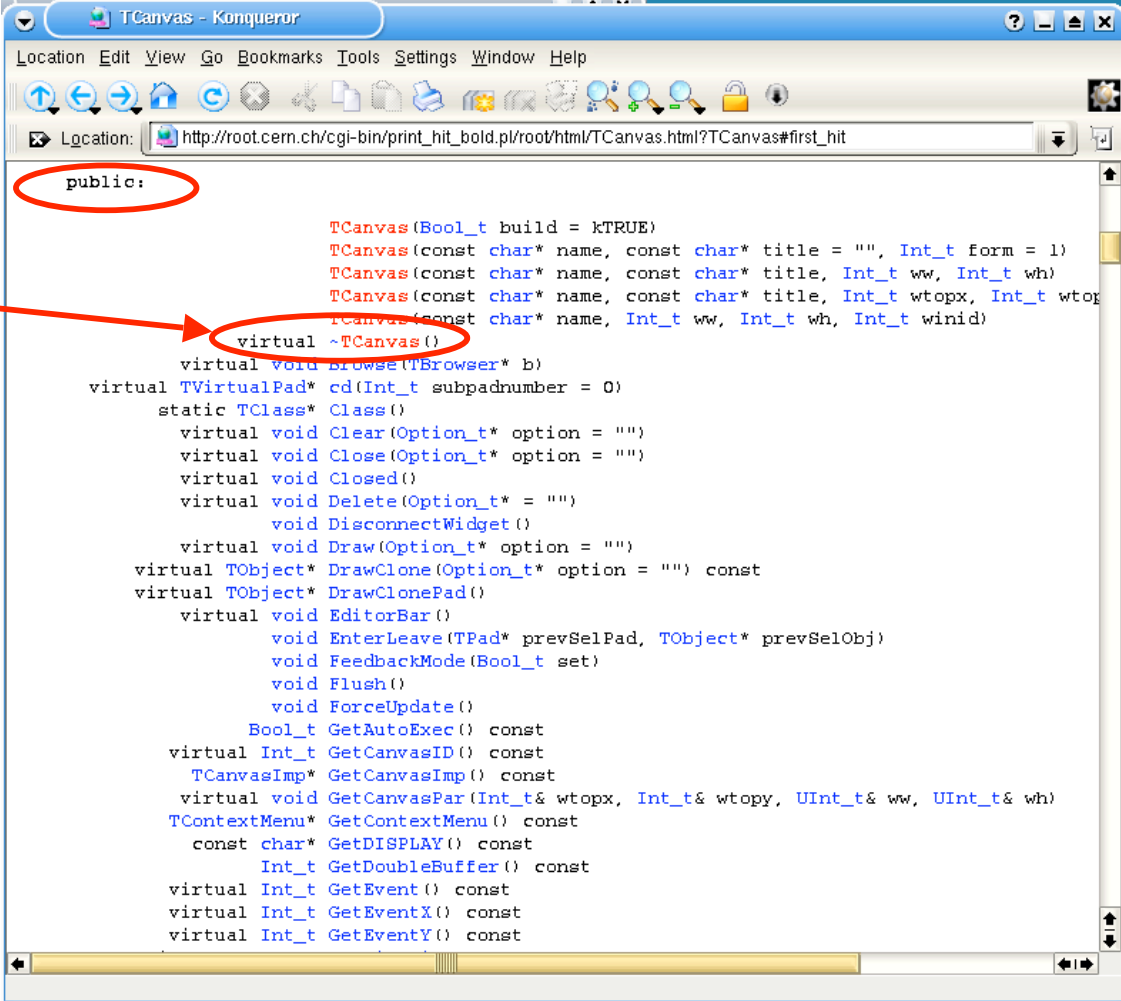
virtual ~TCanvas()
virtual void Browse(TBrowser* b)
virtual TVirtualPad* cd(Int_t subpadnumber = 0)
static TClass* Class()
virtual void Clear(Option_t* option = "")
virtual void Close(Option_t* option = "")
virtual void Closed()
virtual void Delete(Option_t* option = "")
void DisconnectWidget()
virtual void Draw(Option_t* option = "")
virtual TObject* DrawClone(Option_t* option = "") const
virtual TObject* DrawClonePad()
virtual void EditorBar()
void EnterLeave(TPad* prevSelPad, TObject* prevSelObj)
void FeedbackMode(Bool_t set)
void Flush()
void ForceUpdate()
Bool_t GetAutoExec() const
virtual Int_t GetCanvasID() const
TCanvasImp* GetCanvasImp() const
virtual void GetCanvasPar(Int_t& wtopx, Int_t& wtopy, UInt_t& ww, UInt_t& wh)
TContextMenu* GetContextMenu() const
const char* GetDISPLAY() const
Int_t GetDoubleBuffer() const
virtual Int_t GetEvent() const
virtual Int_t GetEventX() const
virtual Int_t GetEventY() const
```

La liste des méthodes d'une classe

- La liste des méthodes "public" est toujours organisée de la même façon:

Une méthode destructeur "`~TypeObjet()`" appelée quand on détruit l'objet, e.g. avec `delete*`

*On ne peut pas utiliser cette méthode directement



```
public:
    TCanvas(Bool_t build = kTRUE)
    TCanvas(const char* name, const char* title = "", Int_t form = 1)
    TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
    TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t ww, Int_t wh)
    TCanvas(const char* name, Int_t ww, Int_t wh, Int_t windid)
    virtual ~TCanvas()
    virtual void Browse(TBrowser* b)
    virtual TVirtualPad* cd(Int_t subpadnumber = 0)
    static TClass* Class()
    virtual void Clear(Option_t* option = "")
    virtual void Close(Option_t* option = "")
    virtual void Closed()
    virtual void Delete(Option_t* option = "")
    void DisconnectWidget()
    virtual void Draw(Option_t* option = "")
    virtual TObject* DrawClone(Option_t* option = "") const
    virtual TObject* DrawClonePad()
    virtual void EditorBar()
    void EnterLeave(TPad* prevSelPad, TObject* prevSelObj)
    void FeedbackMode(Bool_t set)
    void Flush()
    void ForceUpdate()
    Bool_t GetAutoExec() const
    virtual Int_t GetCanvasID() const
    TCanvasImp* GetCanvasImp() const
    virtual void GetCanvasPar(Int_t& wtopx, Int_t& wtopy, UInt_t& ww, UInt_t& wh)
    TContextMenu* GetContextMenu() const
    const char* GetDISPLAY() const
    Int_t GetDoubleBuffer() const
    virtual Int_t GetEvent() const
    virtual Int_t GetEventX() const
    virtual Int_t GetEventY() const
```

Trouver toutes les méthodes d'une classe

- La liste des méthodes "public" est toujours organisée de la même façon:

```
public:

    TCanvas(Bool_t build = kTRUE)
    TCanvas(const char* name, const char* title = "", Int_t form = 1)
    TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
    TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t ww, Int_t wh)
    TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)

    virtual ~TCanvas()

    virtual void Browse(TBrowser* b)
    virtual TVirtualPad* cd(Int_t subpadnumber = 0)
    static TClass* Class()
    virtual void Clear(Option_t* option = "")
    virtual void Close(Option_t* option = "")
    virtual void Closed()
    virtual void Delete(Option_t* option = "")
    void DisconnectWidget()
    virtual void Draw(Option_t* option = "")
    virtual TObject* DrawClone(Option_t* option = "") const
    virtual TObject* DrawClonePad()
    virtual void EditorBar()
    void EnterLeave(TPad* prevSelPad, TObject* prevSelObj)
    void FeedbackMode(Bool_t set)
    void Flush()
    void ForceUpdate()
    Bool_t GetAutoExec() const
    virtual Int_t GetCanvasID() const
    TCanvasImp* GetCanvasImp() const
    virtual void GetCanvasPar(Int_t& wtopx, Int_t& wtopy, UInt_t& ww, UInt_t& wh)
    TContextMenu* GetContextMenu() const
    const char* GetDISPLAY() const
    Int_t GetDoubleBuffer() const
    virtual Int_t GetEvent() const
    virtual Int_t GetEventX() const
    virtual Int_t GetEventY() const
```

La liste alphabétique de toutes les méthodes de la classe...?

Où est la méthode "Divide"?

Trouver toutes les méthodes d'une classe

- Si une classe semble manquer une méthode, elle est peut-être définie par ses (grand-)parents

ASTUCE:

Les objets d'une classe profitent de tout le savoir-faire de leurs aïeux...

...les méthodes passent de mère en fille!!

library: libGpad
#include "TCanvas.h"

TCanvas

[class description](#) - [source file](#) - [inheritance tree](#)

```
class TCanvas : public TPad
```

Inheritance Chart:

- TObject
- TAttLine
- TAttFill <- TVirtualPad <- TPad <- TCanvas <-
- TAttPad
- TQObject

- TAttFillCanvas
- TAttLineCanvas
- TAttMarkerCanvas
- TDialogCanvas <-
- TAttTextCanvas
- TDrawPanelHist
- TFitPanel <- TFitPanelGraph
- TInspectCanvas

```
private:  
  
    TCanvas(const TCanvas& canvas)  
    void Build()  
    virtual void CopyPixmaps()  
    void DrawEventStatus(Int_t event, Int_t x, Int_t y, TObject* selected)  
    TCanvas& operator=(const TCanvas& rhs)  
    void DrawAutoExec()
```

Trouver toutes les méthodes d'une classe

- Si une classe semble manquer une méthode, elle est peut-être définie par ses (grand-)parents

```
public:
    TPad()
    TPad(const char* name, const char* title, Double_t xlow, Double_t xhigh, Double_t ylow, Double_t yhigh)
    virtual ~TPad()
    virtual void AbsCoordinates(Bool_t set)
    virtual Double_t AbsPixeltoX(Int_t px)
    virtual void AbsPixeltoXY(Int_t xpixel, Int_t ypixel, Double_t& x, Double_t& y)
    virtual Double_t AbsPixeltoY(Int_t py)
    virtual void AddExec(const char* name, const char* command)
    virtual void AutoExec()
    virtual void Browse(TBrowser* b)
    virtual TLegend* BuildLegend(Double_t x1 = 0.5, Double_t y1 = 0.67, Double_t x2 = 0.5, Double_t y2 = 0.67)
    virtual TVirtualPad* cd(Int_t subpadnumber = 0)
    static TClass* Class()
    virtual void Clear(Option_t* option = "")
    virtual Int_t Clip(Float_t* x, Float_t* y, Float_t xclip1, Float_t yclip1, Float_t xclip2, Float_t yclip2)
    virtual Int_t Clip(Double_t* x, Double_t* y, Double_t xclip1, Double_t yclip1, Double_t xclip2, Double_t yclip2)
    virtual Int_t ClippingCode(Double_t x, Double_t y, Double_t xc11, Double_t yc11, Double_t xc21, Double_t yc21)
    virtual void Close(Option_t* option = "")
    virtual void Closed()
    virtual void CloseToolTip(TObject* tip)
    virtual void CopyPixmap()
    virtual void CopyPixmaps()
    virtual TObject* CreateToolTip(const TBox* b, const char* text, Long_t delays)
    virtual void DeleteExec(const char* name)
    virtual void DeleteToolTip(TObject* tip)
    virtual void Divide(Int_t nx = 1, Int_t ny = 1, Float_t xmargin = 0.01, Float_t ymargin = 0.01)
    virtual void Draw(Option_t* option = "")
    virtual void DrawClassObject(const TObject* obj, Option_t* option = "")
    static void DrawColorTable()
    virtual void DrawCrosshair()
    virtual TH1F* DrawFrame(Double_t xmin, Double_t ymin, Double_t xmax, Double_t ymax)
    virtual TObject* FindObject(const char* name) const
```

Trouvée!
Regardons la doc...*

*...on expliquera la déclaration (types de variables, arguments par défaut etc.) tout à l'heure

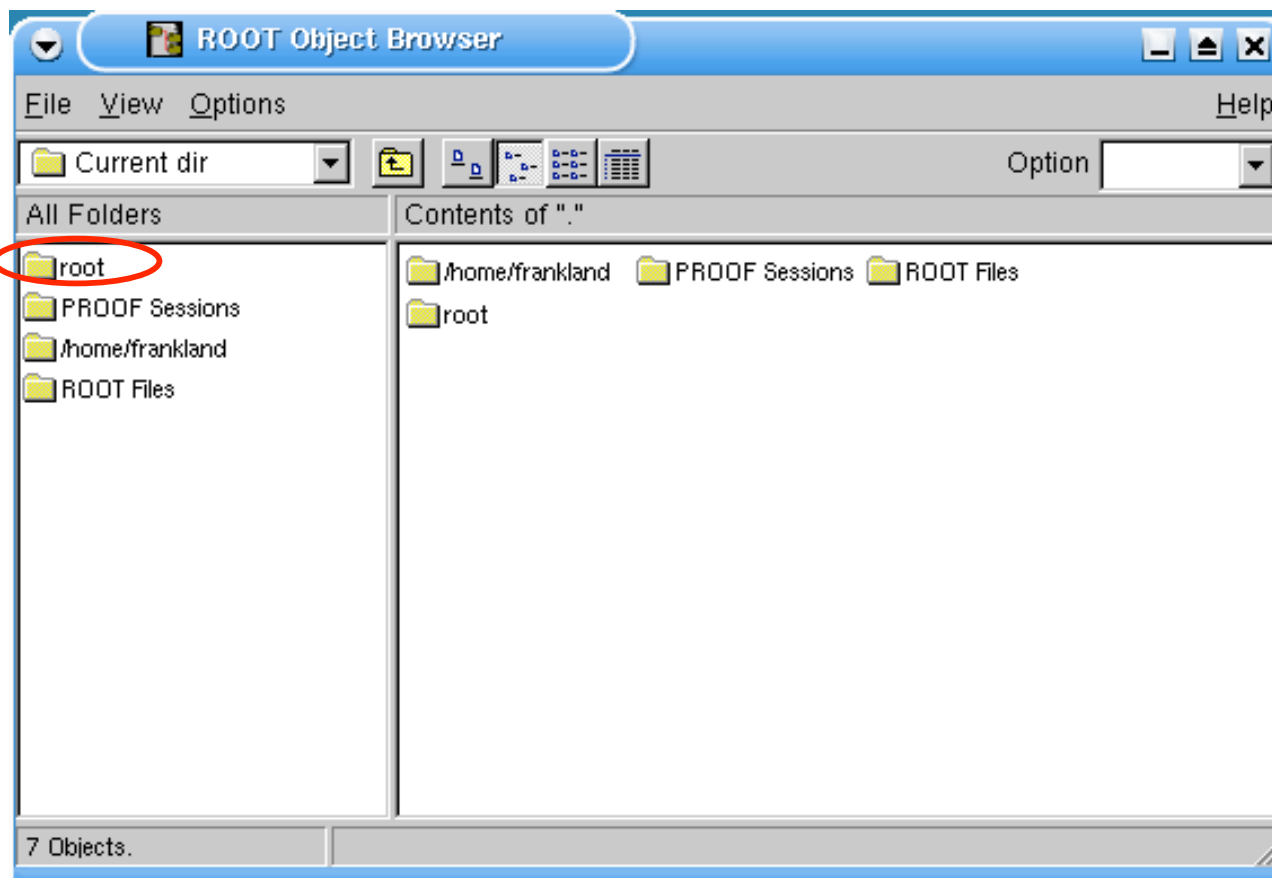
A la recherche des objets perdus

Ou: pourquoi les objets ont des noms

Retrouver un objet perdu

- ROOT tient à jour des listes d'objets, ce qui permet de les retrouver facilement

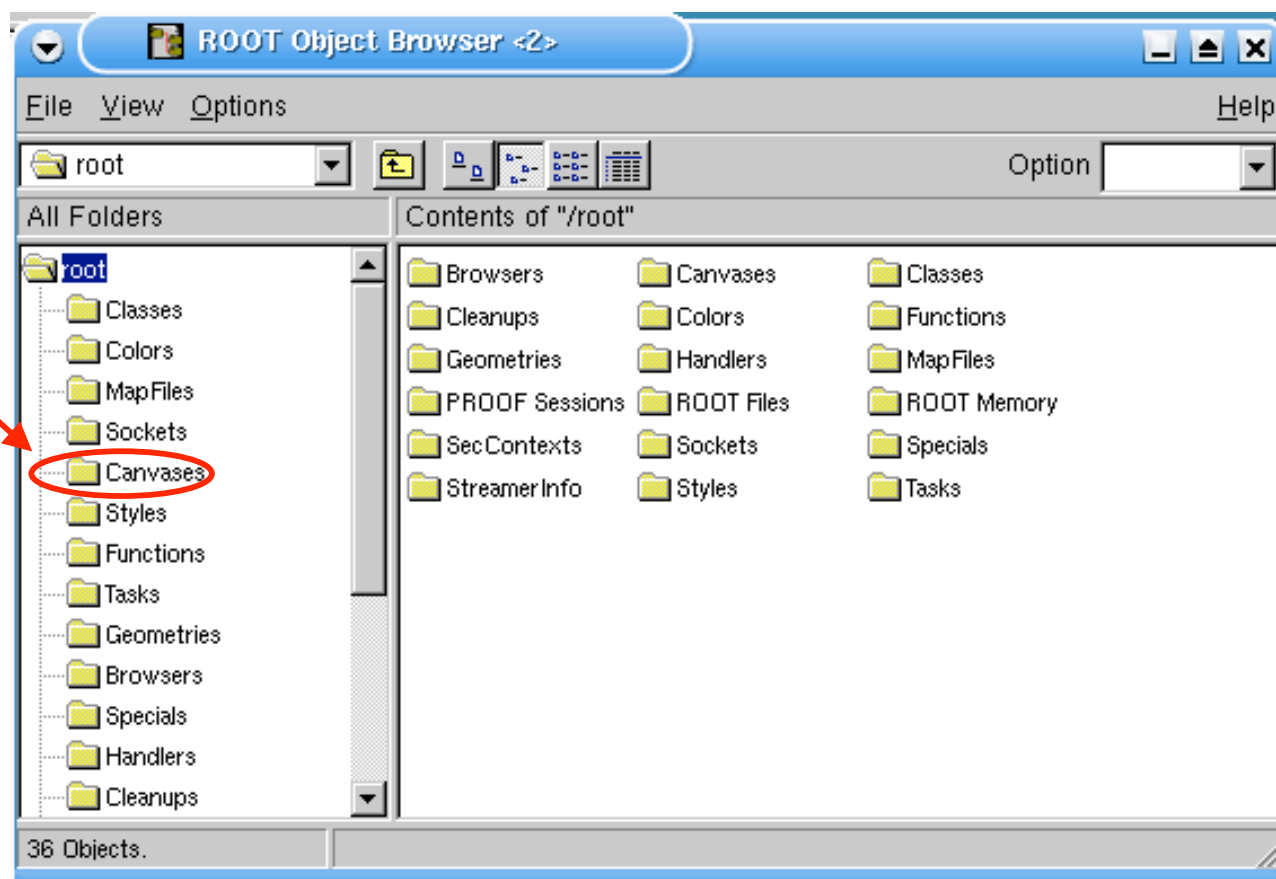
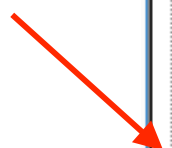
Double-
clique sur
"root"



Retrouver un objet perdu

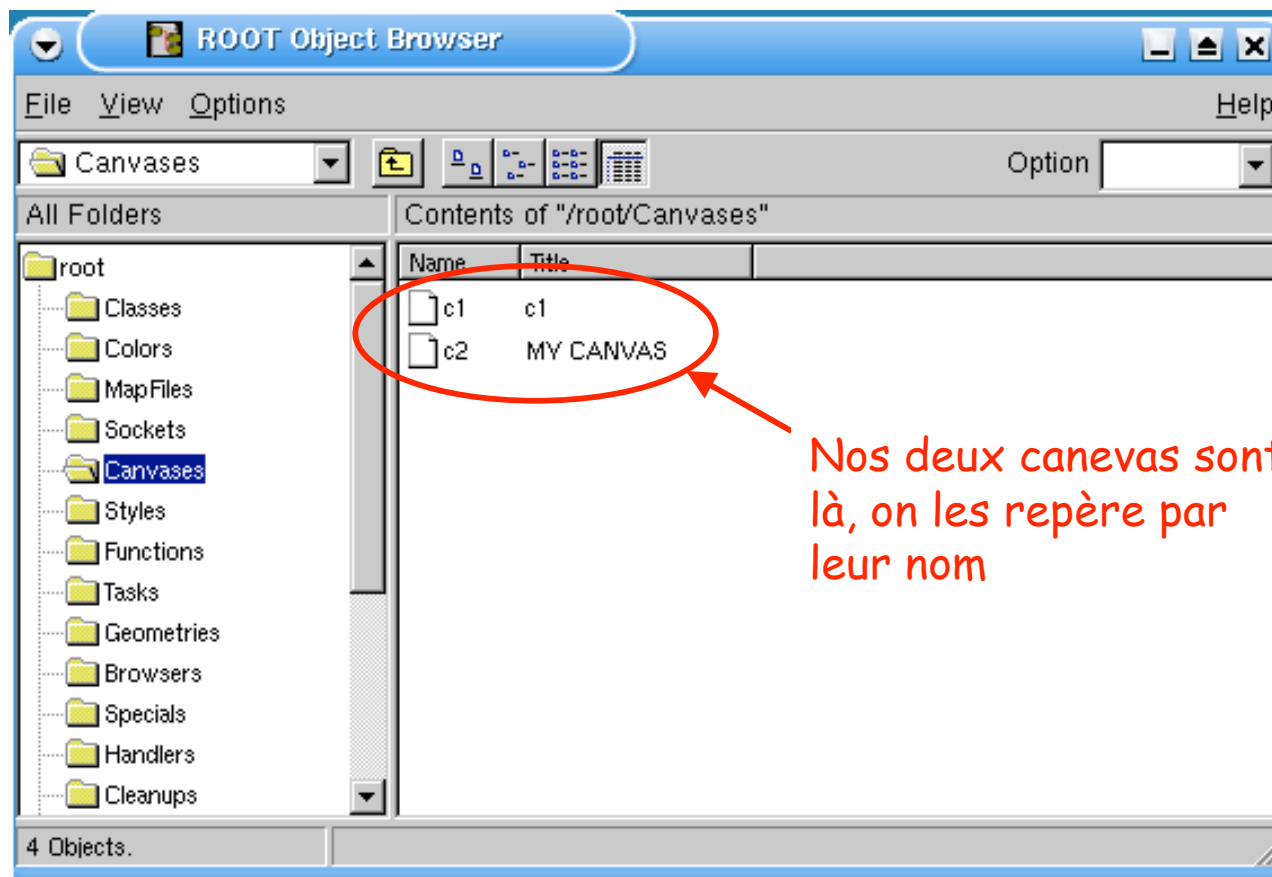
- On peut naviguer dans ces listes en utilisant le TBrowser

Clique sur
"Canvases"



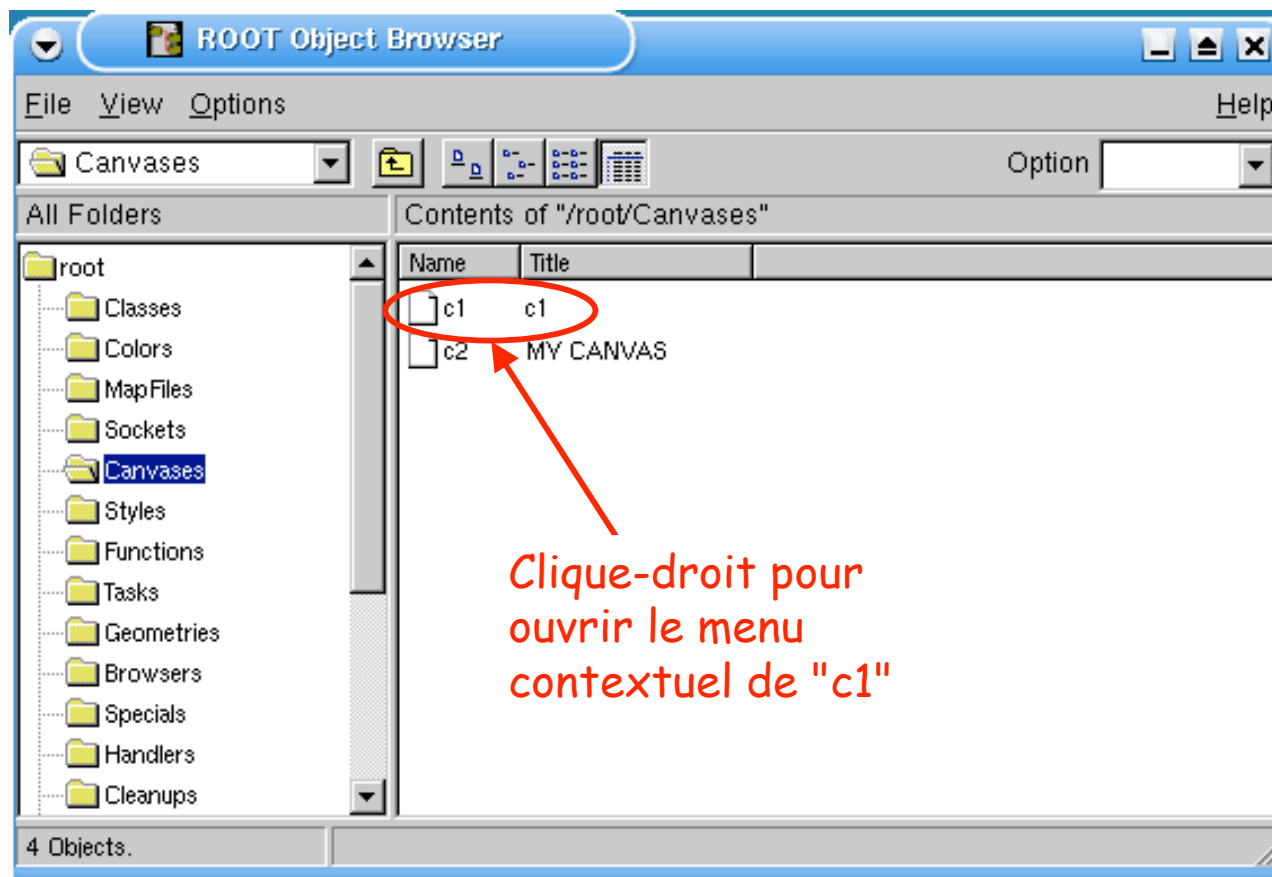
Retrouver un objet perdu

- Chaque objet doit porter un nom unique, pour qu'on puisse le retrouver



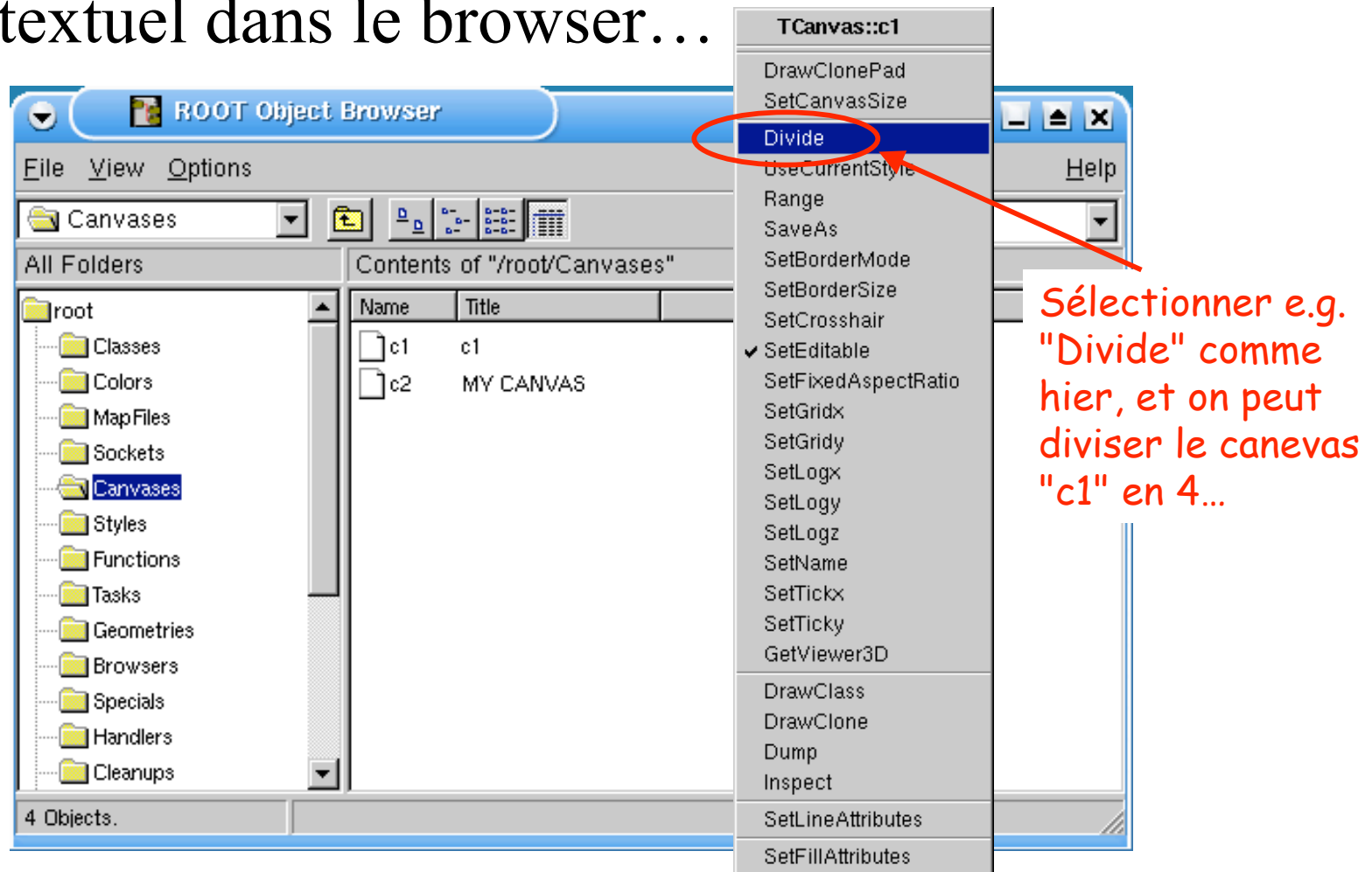
Retrouver un objet perdu

- On peut agir sur un objet à travers son menu contextuel dans le browser...



Retrouver un objet perdu

- On peut agir sur un objet à travers son menu contextuel dans le browser...



Retrouver un objet perdu

- ...ou en récupérant son adresse, on agira avec un pointeur d'objet:

```
gROOT->GetListOfCanvases()->FindObject("c1");
```

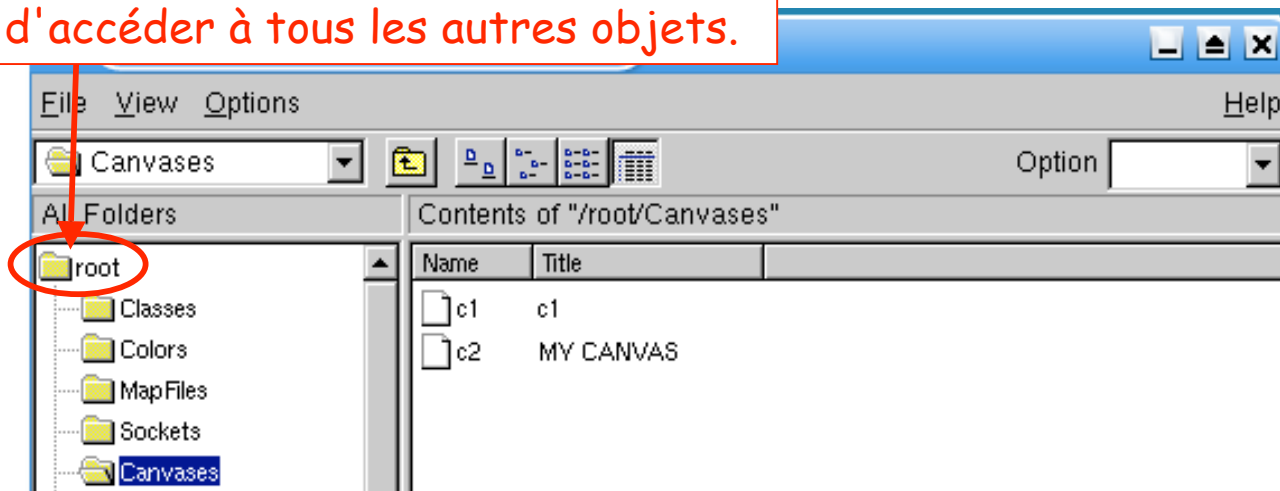
Retrouver un objet perdu

- ...ou en récupérant son adresse, on agira avec un pointeur d'objet:

```
gROOT->GetListOfCanvases()->FindObject("c1");
```

Ce pointeur d'objet contient l'adresse du dossier racine "root" dans le TBrowser.

Il permet d'accéder à tous les autres objets.

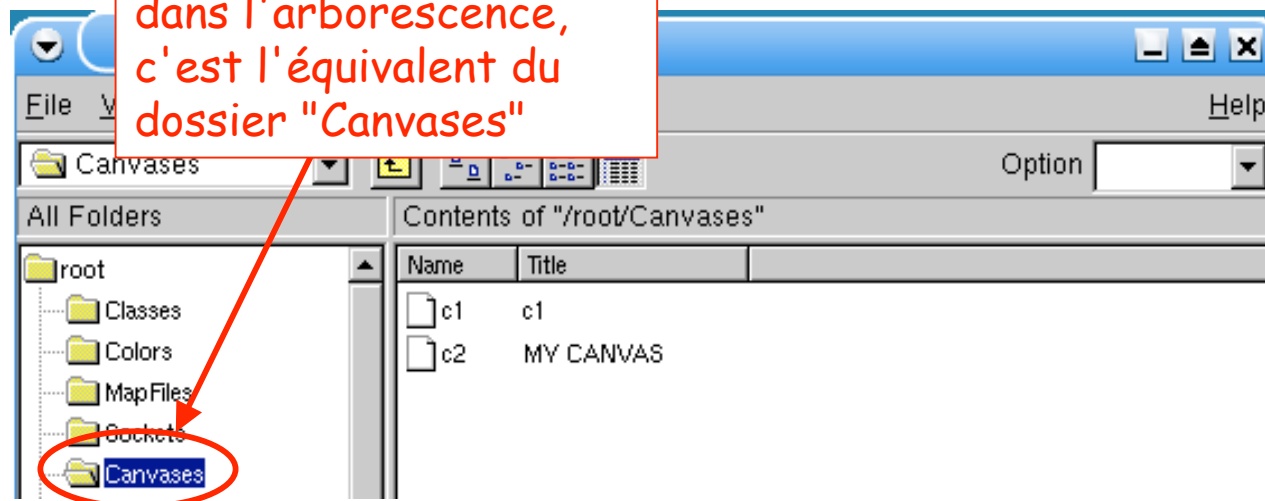


Retrouver un objet perdu

- ...ou en récupérant son adresse, on agira avec un pointeur d'objet:

```
gROOT->GetListOfCanvases()->FindObject("c1");
```

On descend d'un niveau dans l'arborescence, c'est l'équivalent du dossier "Canvases"

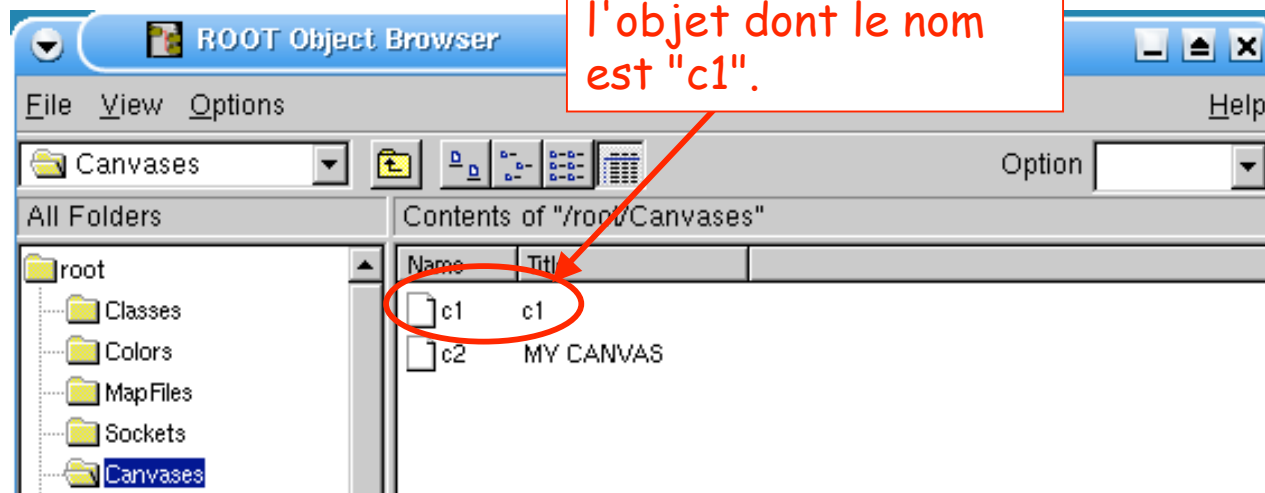


Retrouver un objet perdu

- ...ou en récupérant son adresse, on agira avec un pointeur d'objet:

```
gROOT->GetListOfCanvases()->FindObject("c1");
```

Dans la liste des canevas, on cherche l'objet dont le nom est "c1".



Retrouver un objet perdu

- Si l'on n'est pas sûr dans quel dossier il faut chercher, on peut faire une recherche récursive dans tous les dossiers:

```
gROOT->GetListOfCanvases()->FindObject("c1");
```

```
gROOT->FindObject("nom");
```

C'est la formule magique qui permet de retrouver à peu près n'importe quel objet à n'importe quel moment.

On l'utilisera tout le temps!!

Retrouver un objet perdu

- Ensuite il n'y a plus qu'à mettre l'adresse dans un pointeur approprié et l'utiliser:

```
TypeObjet* toto = (TypeObjet*) gROOT->FindObject("nom");
```

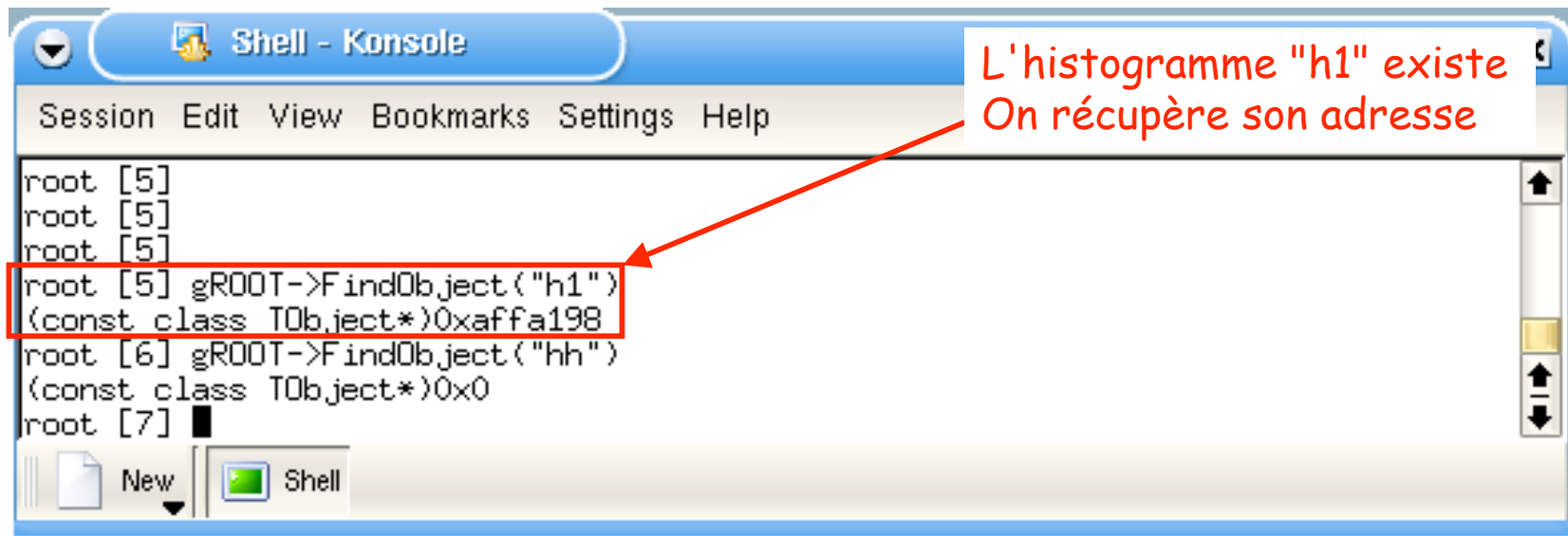
On doit préciser
le type de l'objet
recherché ici

Exemple: on cherche le
canevas "c1" et on efface
son contenu:

```
TCanvas* c1_ptr = (TCanvas*) gROOT->FindObject("c1");  
c1_ptr->Clear();
```

Savoir si un objet existe

- On utilisera la même fonction quand on voudra savoir si tel ou tel objet a déjà été créé



The screenshot shows a Qt Shell console window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The console output shows the following commands and results:

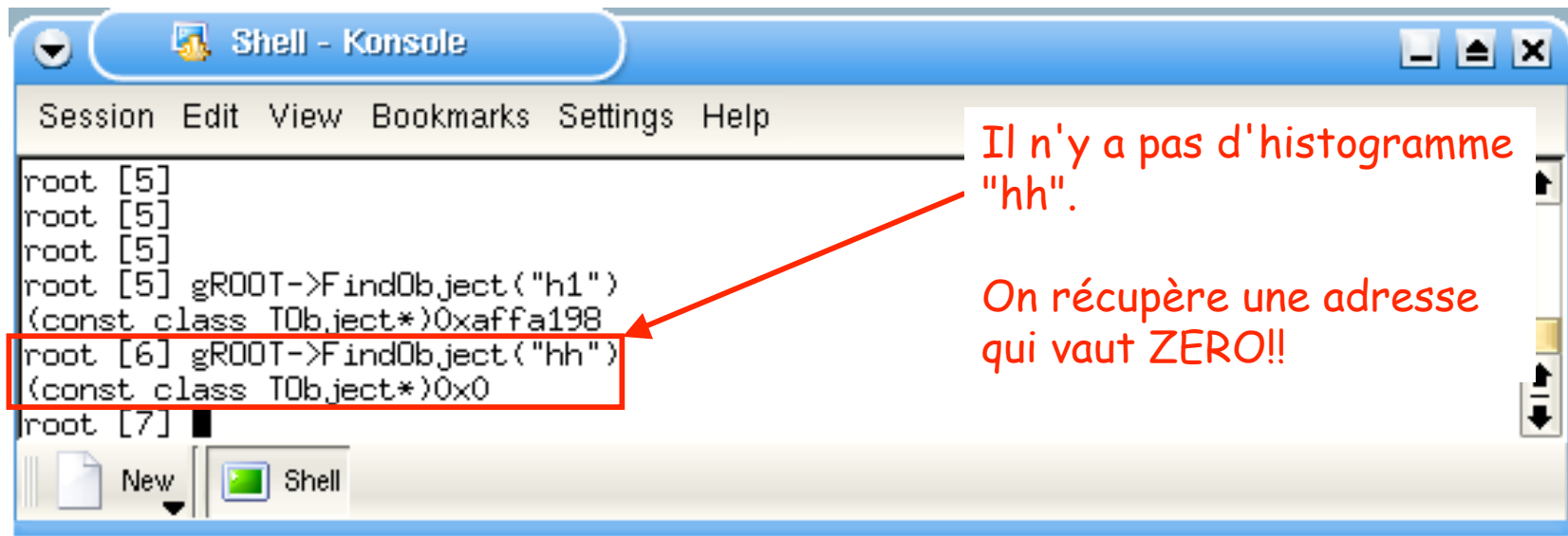
```
root [5]
root [5]
root [5]
root [5] gROOT->FindObject("h1")
(const class TObject*)0xaffa198
root [6] gROOT->FindObject("hh")
(const class TObject*)0x0
root [7] █
```

A red box highlights the output of the first command: `(const class TObject*)0xaffa198`. A red arrow points from a text box to this output. The text box contains the following text:

L'histogramme "h1" existe
On récupère son adresse

Savoir si un objet existe

- On utilisera la même fonction quand on voudra savoir si tel ou tel objet a déjà été créé



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
root [5]
root [5]
root [5]
root [5] gROOT->FindObject("h1")
(const class TObject*)0xaffa198
root [6] gROOT->FindObject("hh")
(const class TObject*)0x0
root [7]
```

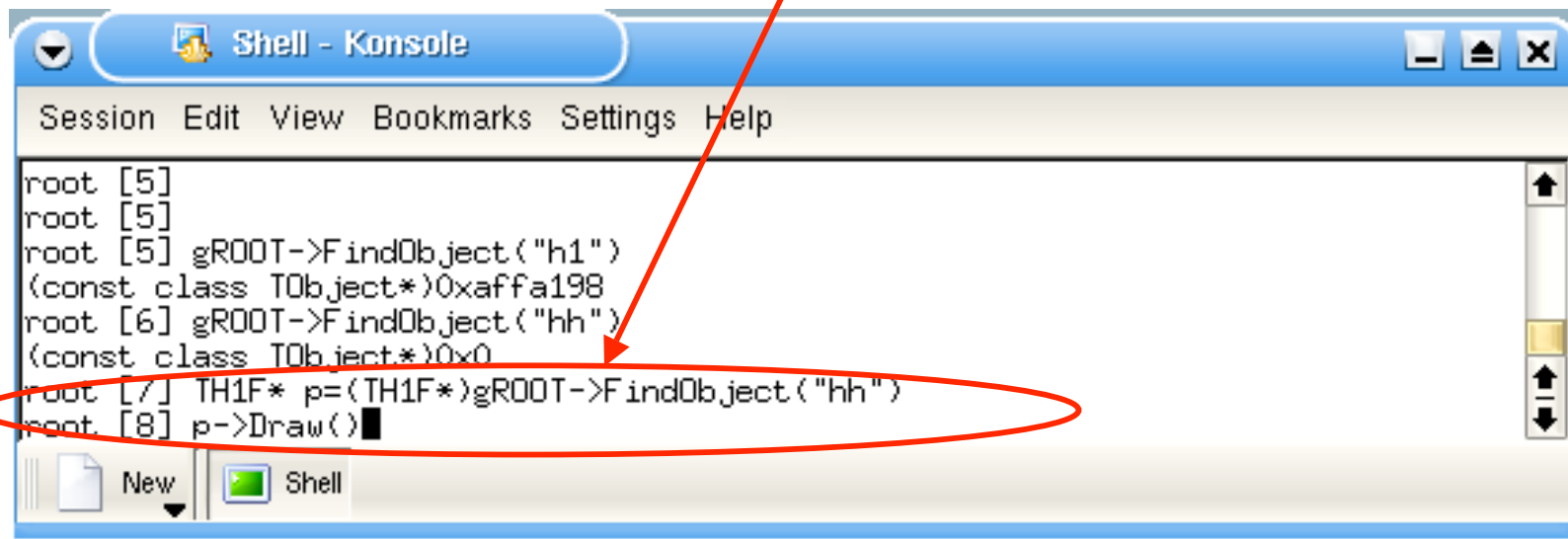
Il n'y a pas d'histogramme "hh".

On récupère une adresse qui vaut ZERO!!

Savoir si un objet existe

- En toute rigueur, il faut toujours tester la valeur d'un pointeur pour être sûr que l'adresse est valable...

Qu'est-ce qui se passe quand on essaie d'utiliser un pointeur nul ?

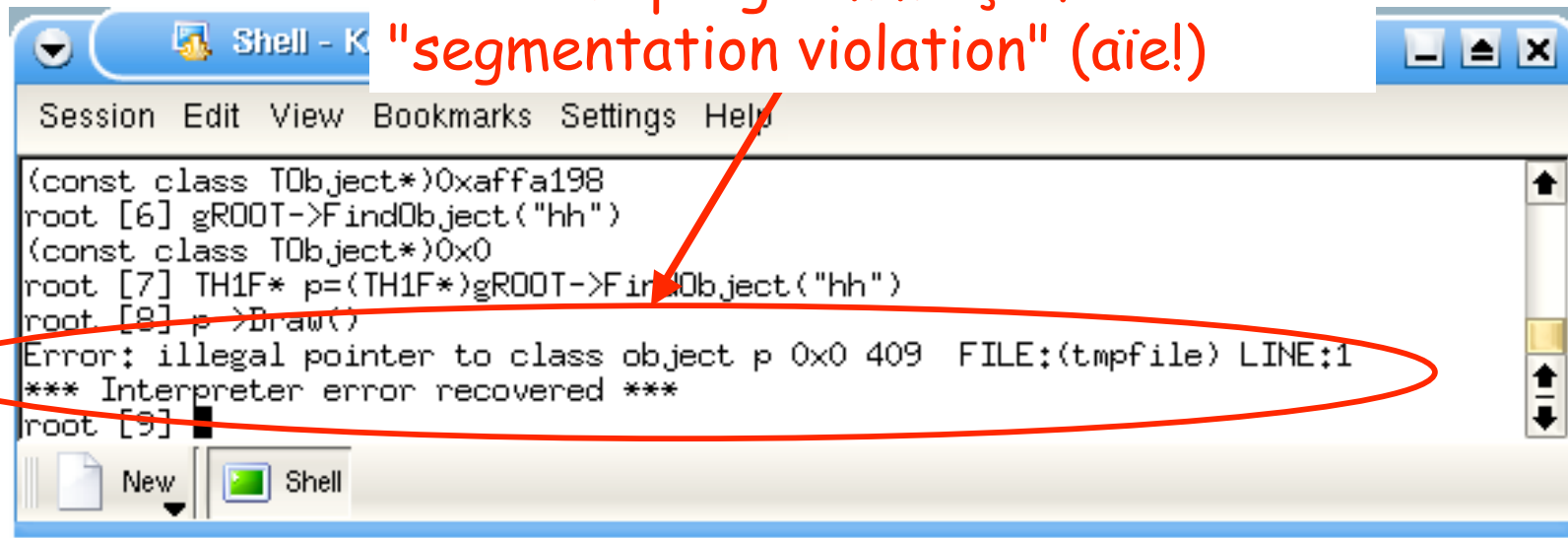


```
Shell - Konsole
Session Edit View Bookmarks Settings Help
root [5]
root [5]
root [5] gROOT->FindObject("h1")
(const class TObject*)0xaffa198
root [6] gROOT->FindObject("hh")
(const class TObject*)0x0
root [7] TH1F* p=(TH1F*)gROOT->FindObject("hh")
root [8] p->Draw()
```

Savoir si un objet existe

- En toute rigueur, il faut toujours tester la valeur d'un pointeur pour être sûr que l'adresse est valable...

L'interpréteur est gentil...
...dans un programme ça ferait
"segmentation violation" (aie!)



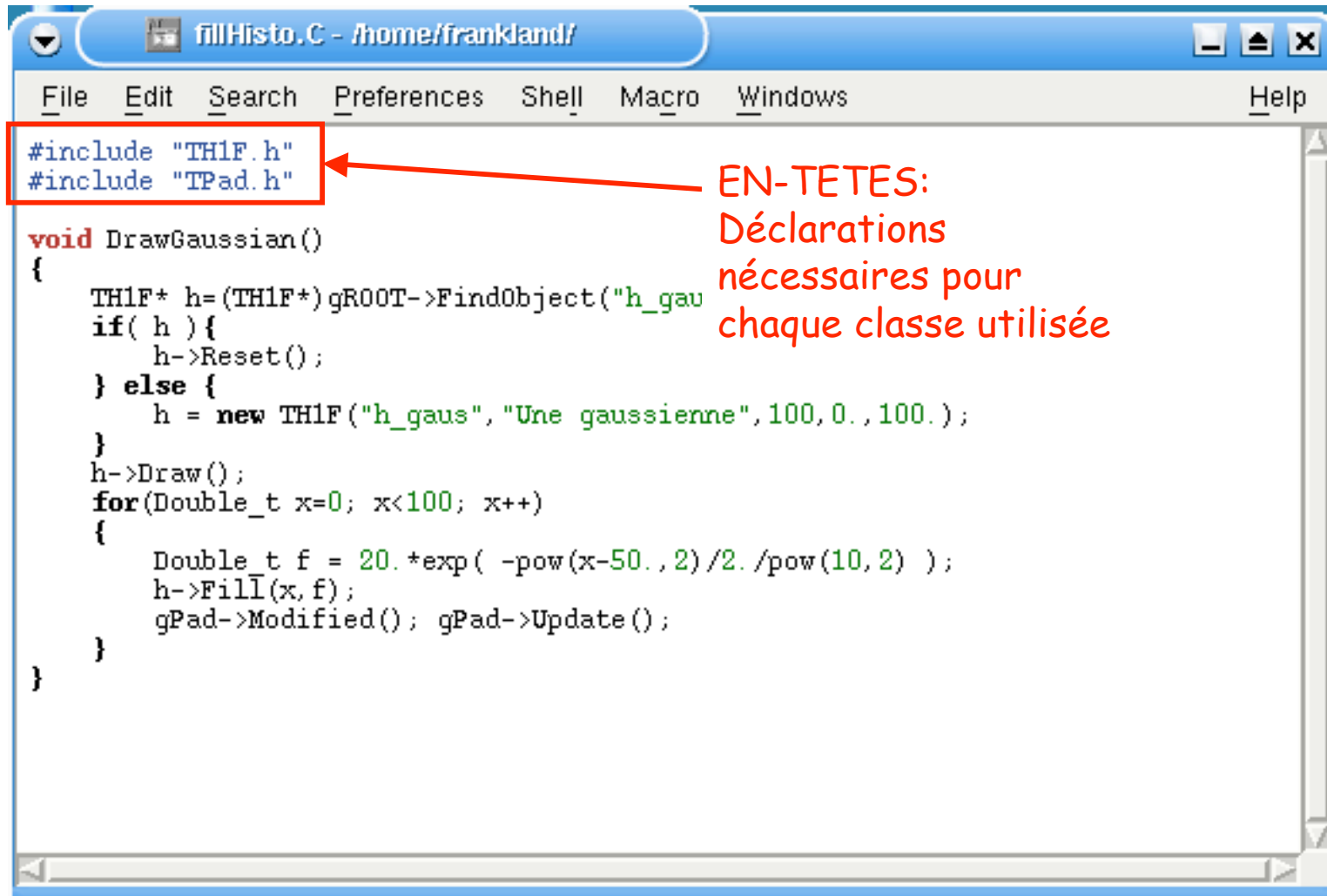
```
Shell - K...
Session Edit View Bookmarks Settings Help
(const class Tobject*)0xaffa198
root [6] gROOT->FindObject("hh")
(const class Tobject*)0x0
root [7] TH1F* p=(TH1F*)gROOT->FindObject("hh")
root [8] p->Draw()
Error: illegal pointer to class object p 0x0 409 FILE:(tmpfile) LINE:1
*** Interpreter error recovered ***
root [9]
```

Ecriture de fonctions

La programmation C++ par les nuls

Une fonction

- Voici un exemple d'une fonction C++



The image shows a screenshot of a C++ code editor window titled "fillHisto.C - /home/frankland/". The window has a menu bar with "File", "Edit", "Search", "Preferences", "Shell", "Macro", "Windows", and "Help". The code is as follows:

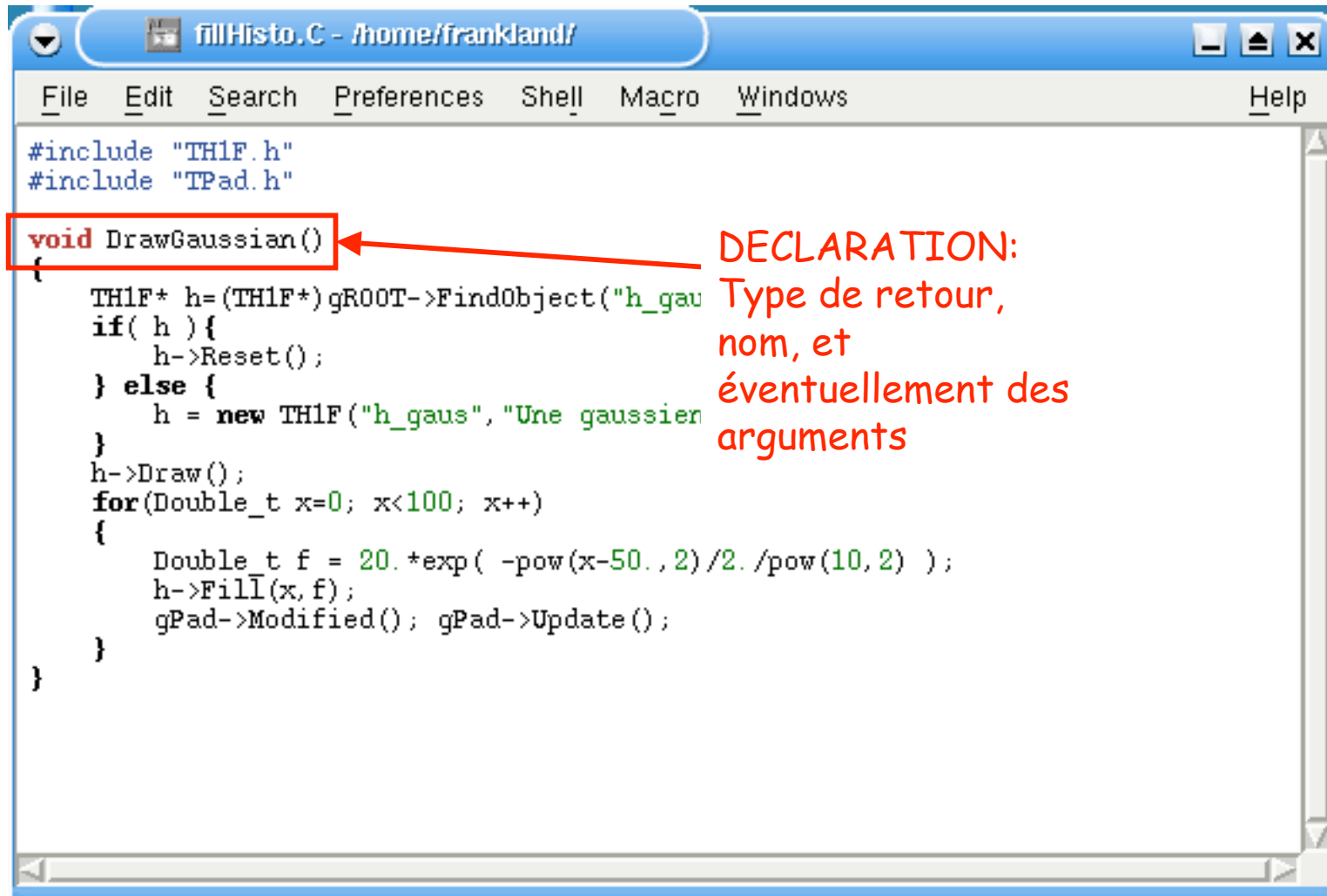
```
#include "TH1F.h"
#include "TPad.h"

void DrawGaussian()
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gau
    if( h ){
        h->Reset();
    } else {
        h = new TH1F("h_gaus", "Une gaussienne",100,0.,100.);
    }
    h->Draw();
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = 20.*exp( -pow(x-50.,2)/2./pow(10,2) );
        h->Fill(x, f);
        gPad->Modified(); gPad->Update();
    }
}
```

An orange arrow points from the text "EN-TETES: Déclarations nécessaires pour chaque classe utilisée" to the two include lines at the top of the code.

Une fonction

- Voici un exemple d'une fonction C++

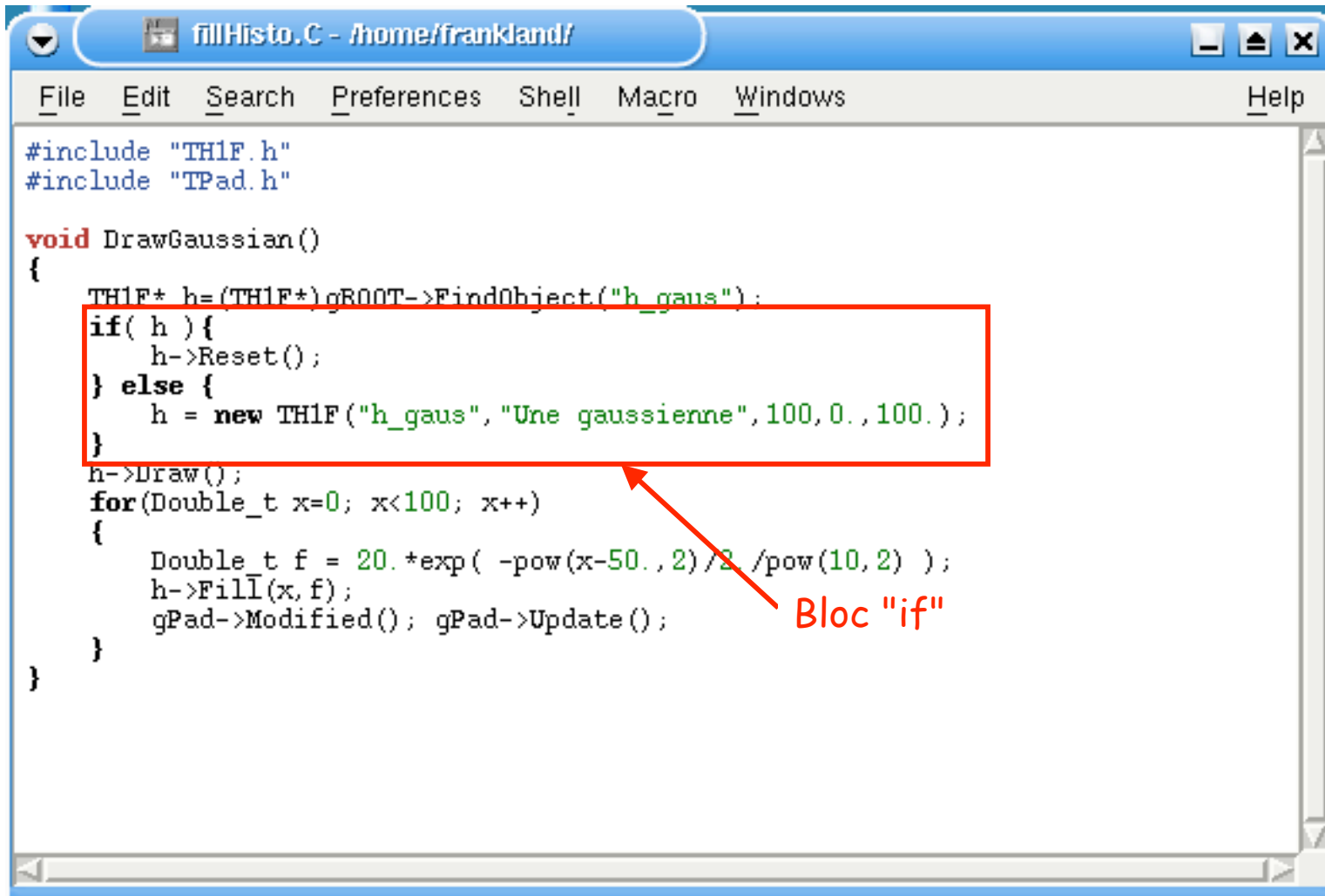


```
fillHisto.C - /home/frankland/  
File Edit Search Preferences Shell Macro Windows Help  
#include "TH1F.h"  
#include "TPad.h"  
void DrawGaussian()  
{  
    TH1F* h=(TH1F*)gROOT->FindObject("h_gau  
    if( h ){  
        h->Reset();  
    } else {  
        h = new TH1F("h_gaus", "Une gaussien  
    }  
    h->Draw();  
    for(Double_t x=0; x<100; x++)  
    {  
        Double_t f = 20.*exp( -pow(x-50., 2)/2. /pow(10, 2) );  
        h->Fill(x, f);  
        gPad->Modified(); gPad->Update();  
    }  
}
```

DECLARATION:
Type de retour,
nom, et
éventuellement des
arguments

Une fonction

- Voici un exemple d'une fonction C++

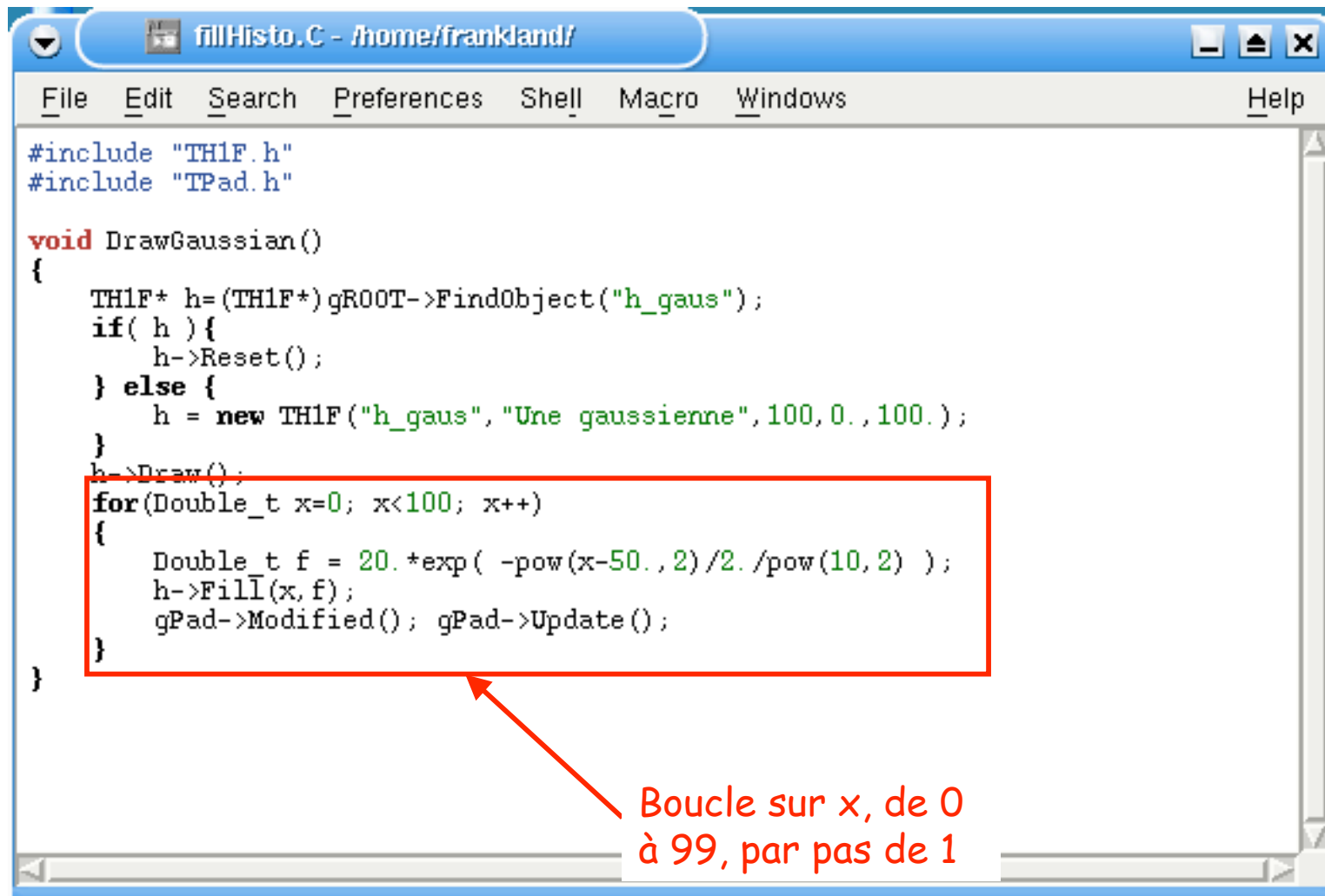


```
fillHisto.C - /home/frankland/  
File Edit Search Preferences Shell Macro Windows Help  
#include "TH1F.h"  
#include "TPad.h"  
  
void DrawGaussian()  
{  
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");  
    if( h ){  
        h->Reset();  
    } else {  
        h = new TH1F("h_gaus", "Une gaussienne",100,0.,100.);  
    }  
    h->Draw();  
    for(Double_t x=0; x<100; x++)  
    {  
        Double_t f = 20.*exp( -pow(x-50.,2)/2 /pow(10,2) );  
        h->Fill(x, f);  
        gPad->Modified(); gPad->Update();  
    }  
}
```

Bloc "if"

Une fonction

- Voici un exemple d'une fonction C++



```
fillHisto.C - /home/frankland/
File Edit Search Preferences Shell Macro Windows Help

#include "TH1F.h"
#include "TPad.h"

void DrawGaussian()
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");
    if( h ){
        h->Reset();
    } else {
        h = new TH1F("h_gaus", "Une gaussienne", 100, 0., 100.);
    }
    h->Draw();
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = 20.*exp( -pow(x-50., 2)/2. /pow(10, 2) );
        h->Fill(x, f);
        gPad->Modified(); gPad->Update();
    }
}
```

Boucle sur x, de 0 à 99, par pas de 1

Une fonction

- Voici un exemple d'une fonction C++

```
fillHisto.C - /home/frankland/
File Edit Search Preferences Shell
#include "TH1F.h"
#include "TPad.h"

void DrawGaussian()
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");
    if( h ){
        h->Reset();
    } else {
        h = new TH1F("h_gaus", "Une gaussienne",100,0.,100.);
    }
    h->Draw();
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = 20.*exp( -pow(x-50.,2)/2./pow(10,2) );
        h->Fill(x,f);
        gPad->Modified(); gPad->Update();
    }
}
```

Mettre à zéro le contenu du spectre

gPad = pointeur global du pad/canevas actif

Compilation et utilisation

- On compile et on charge la fonction:

`.L fillHisto.C+`

Nom du fichier qui
contient la fonction



- On exécute la fonction:

`DrawGaussian()`

Nom de la fonction



Pourquoi utiliser "new" pour créer l'histo ?

- Voyons ce qui se passe si on crée notre histogramme sans utiliser "new"

The screenshot shows a code editor window titled "fillHistoLocal.C - /home/frankland/". The code defines a function `DrawGaussian0()` that creates a `TH1F` object `h` and fills it with a Gaussian distribution. The code is as follows:

```
#include "TH1F.h"
#include "TPad.h"

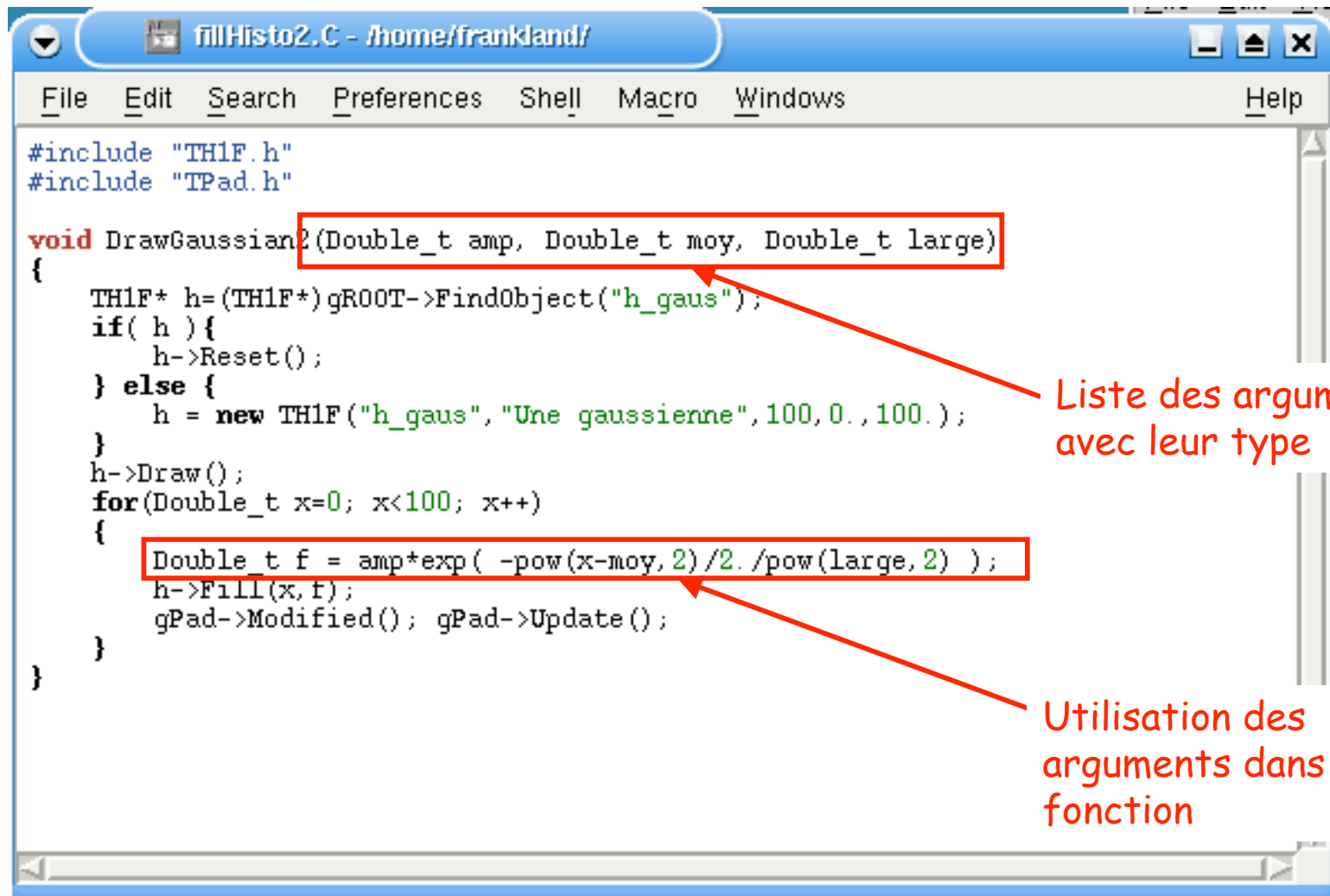
void DrawGaussian0()
{
    TH1F h("h_gaus", "Une gaussienne", 100, 0., 100.);
    h.Draw();
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = 20.*exp(-pow(x-50., 2)/2./pow(10, 2));
        h.Fill(x, f);
        gPad->Modified(); gPad->Update();
    }
}
```

Three red arrows point to specific parts of the code with the following annotations:

- An arrow points to the line `TH1F h("h_gaus", "Une gaussienne", 100, 0., 100.);` with the text: "Objet temporaire, il n'existe que dans ce bloc (fonction)".
- An arrow points to the `h.Fill(x, f);` line inside the loop with the text: "On voit l'histo se remplir...".
- An arrow points to the closing brace `}` of the function with the text: "... puis disparaître à la fin de la fonction".

Comment utiliser les arguments

- Une fonction avec des arguments:



```
fillHisto2.C - /home/frankland/
File Edit Search Preferences Shell Macro Windows Help

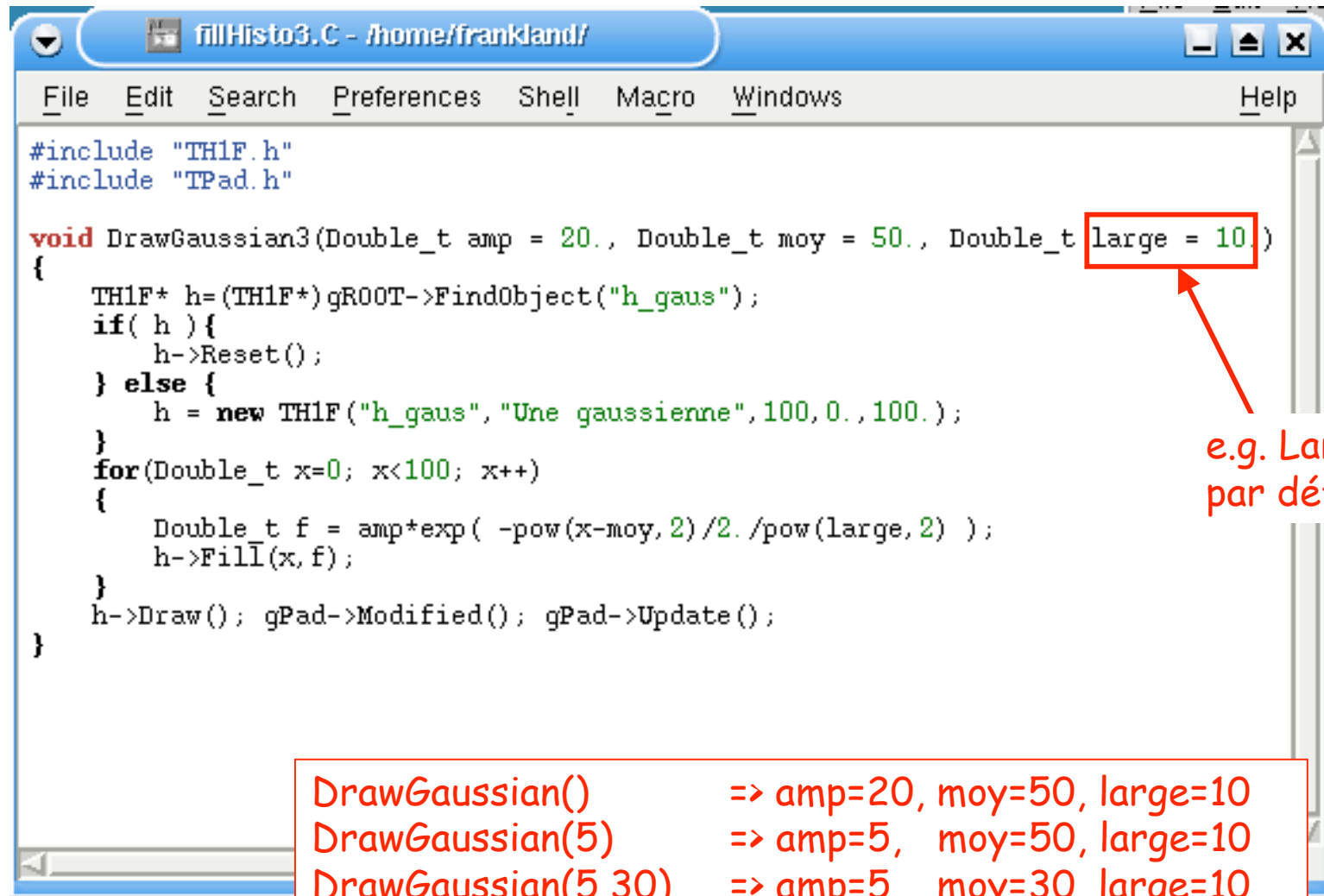
#include "TH1F.h"
#include "TPad.h"

void DrawGaussian2(Double_t amp, Double_t moy, Double_t large)
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");
    if( h ){
        h->Reset();
    } else {
        h = new TH1F("h_gaus", "Une gaussienne", 100, 0., 100.);
    }
    h->Draw();
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = amp*exp( -pow(x-moy,2)/2./pow(large,2) );
        h->Fill(x, f);
        gPad->Modified(); gPad->Update();
    }
}
```

Liste des arguments avec leur type

Utilisation des arguments dans la fonction

Valeurs d'arguments par défaut



```
#include "TH1F.h"
#include "TPad.h"

void DrawGaussian3(Double_t amp = 20., Double_t moy = 50., Double_t large = 10.)
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");
    if( h ){
        h->Reset();
    } else {
        h = new TH1F("h_gaus", "Une gaussienne",100,0.,100.);
    }
    for(Double_t x=0; x<100; x++)
    {
        Double_t f = amp*exp( -pow(x-moy,2)/2./pow(large,2) );
        h->Fill(x, f);
    }
    h->Draw(); gPad->Modified(); gPad->Update();
}
```

e.g. Largeur
par défaut

<code>DrawGaussian()</code>	\Rightarrow amp=20, moy=50, large=10
<code>DrawGaussian(5)</code>	\Rightarrow amp=5, moy=50, large=10
<code>DrawGaussian(5,30)</code>	\Rightarrow amp=5, moy=30, large=10
<code>DrawGaussian(5,30,5)</code>	\Rightarrow amp=5, moy=30, large=5

Retour de valeurs/objets

- Les fonctions peuvent retourner tous les types de variables

```
fillHisto4.C - /home/fran...
File Edit Search Preferences Windows Help

#include "TH1F.h"
#include "TPad.h"

Double_t Gaussian(Double_t z, Double_t toto, Double_t tata, Double_t tutu)
{
    Double_t f = toto*exp( -pow(z-tata,2)/2./pow(tutu,2) );
    return f;
}

TH1F* DrawGaussian4(Double_t amp = 20., Double_t moy = 50., Double_t large = 10.)
{
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");
    if( h ){
        h->Reset();
        new TH1F("h_gaus", "Une gaussienne", 100,
        le_t x=0; x<100; x++)
        Double_t f = Gaussian(x, amp, moy, large);
        h->Fill(x, f);
    }
    return h;
}
```

Type de la variable retournée

Renvoi de la valeur calculée

Fonction gaussienne

Retour de valeurs/objets

- Les fonctions peuvent retourner tous les types de variables

```
fillHisto4.C - /home/frankland/  
File Edit Search Preferences Shell Macro Windows Help  
#include "TH1F.h"  
#include "TROOT.h"  
Double_t G: toto, Double_t tata, Double_t tutu)  
{  
    Double_t r = (toto + tata + tutu) / 3.0;  
    return r;  
}  
TH1F* DrawGaussian4(Double_t amp = 20., Double_t moy = 50., Double_t large = 10.)  
{  
    TH1F* h=(TH1F*)gROOT->FindObject("h_gaus");  
    if( h ){  
        h->Reset();  
    } else {  
        h = new TH1F("h_gaus", "Une gaussienne", 100, 0., 100.);  
    }  
    for(Double_t x=0; x<100; x++)  
    {  
        Double_t f = Gaussian(x, amp, moy, large);  
        h->Fill(x, f);  
    }  
    return h;  
}
```

On retourne un pointeur d'histogramme

Utilisation de la fonction gaussienne

Renvoi du pointeur

Retour de valeurs/objets

- Utilisation de la fonction

On peut utiliser la fonction gaussienne indépendamment

```
Shell - Konsole
Session Edit View Bookmarks Settings Help
root [0] ./L fillHisto4.C+
Info in <TUnixSystem::ACLiC>: creating shared library /home/frankland/./fillHisto4_C.dll
root [1] Gaussian(4,3,2,1)
(Double_t)4.06005849709838107e-01
root [2] IH1F* qq=DrawGaussian4()
root [3] qq->Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [4] █
```

On exécute la fonction, récupère le pointeur de l'histogramme, et on l'affiche...

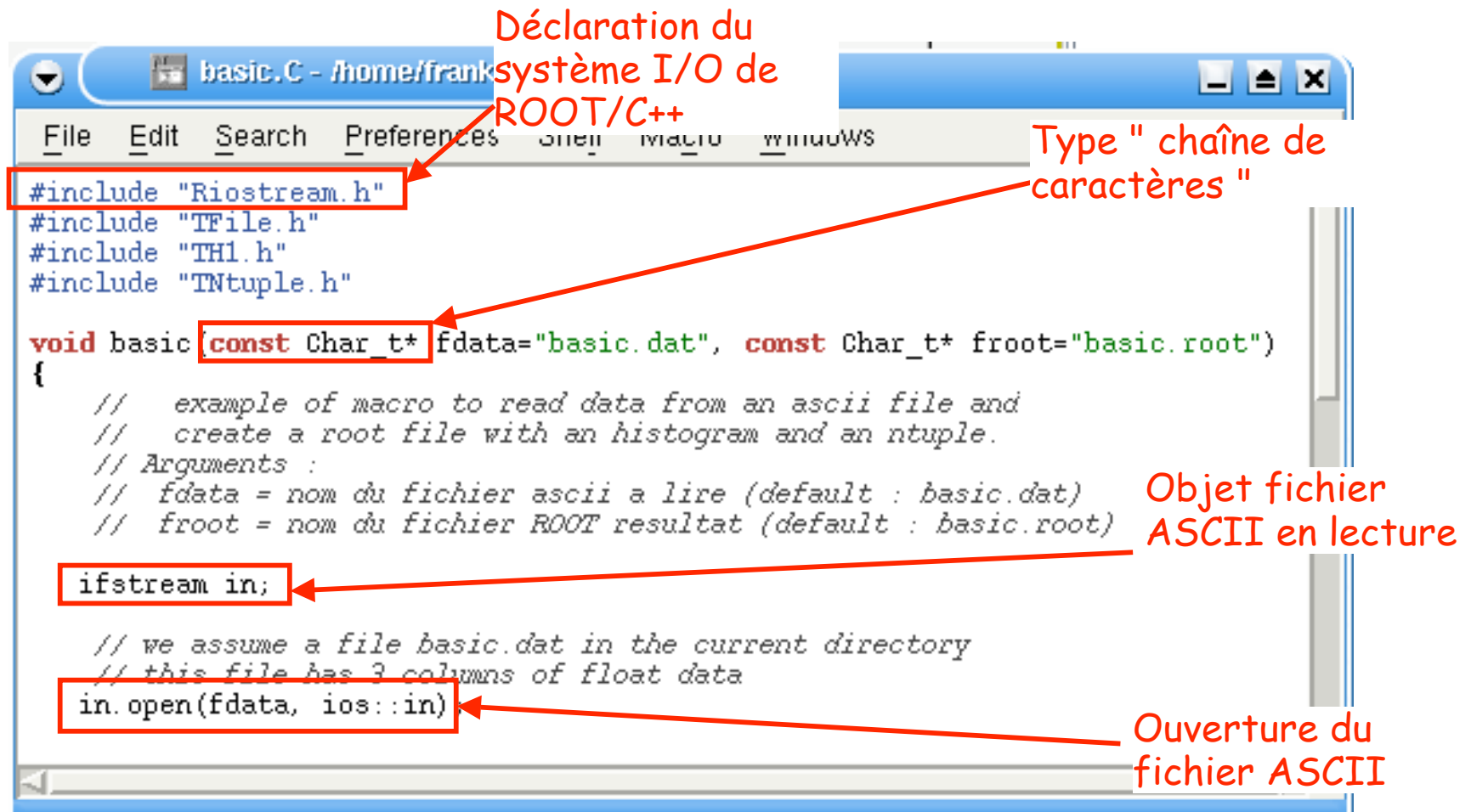
Exemple d'analyse

<http://caeinfo.in2p3.fr/root/Formation/fr/Jour2/basic.C>

<http://caeinfo.in2p3.fr/root/Formation/fr/Jour2/basic.dat>

Exemple d'analyse

Un exemple de script d'analyse: on lit les données dans un fichier ASCII `basic.dat`; on génère quelques histogrammes que l'on va sauvegarder dans le fichier `basic.root`.



```
basic.C - /home/frank
File Edit Search Preferences
#include "Riostream.h"
#include "TFile.h"
#include "TH1.h"
#include "TNTuple.h"

void basic(const Char_t* fdata="basic.dat", const Char_t* froot="basic.root")
{
    // example of macro to read data from an ascii file and
    // create a root file with an histogram and an ntuple.
    // Arguments :
    // fdata = nom du fichier ascii a lire (default : basic.dat)
    // froot = nom du fichier ROOT resultat (default : basic.root)

    ifstream in;

    // we assume a file basic.dat in the current directory
    // this file has 3 columns of float data

    in.open(fdata, ios::in);
}
```

Déclaration du système I/O de ROOT/C++

Type " chaîne de caractères "

Objet fichier ASCII en lecture

Ouverture du fichier ASCII

Exemple d'analyse

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

Déclarations de quelques variables

Création de nouveau fichier ROOT, on l'écrase s'il existe déjà.

Création de Ntuple

Les histogrammes etc. créés après l'ouverture du fichier sont associés à ce fichier; ils lui appartiennent.

Exemple d'analyse

Boucle "while": on continue tant que la lecture de fichier ASCII est correcte

```
basic.C - /home/frankland/
File Edit Search Preferences Shell Ma Help
while (in.good())
{
  in >> x >> y >> z;
  if (in.good())
  {
    if (nlines < 5)
    {
      cout << "X = " << x << ", Y = " << y ;
      cout << ", Z = " << z << endl;
    }
    hl->Fill(x);
    ntuple->Fill(x,y,z);
    nlines++;
  }
}
```

Lecture des 3 paramètres dans le fichier

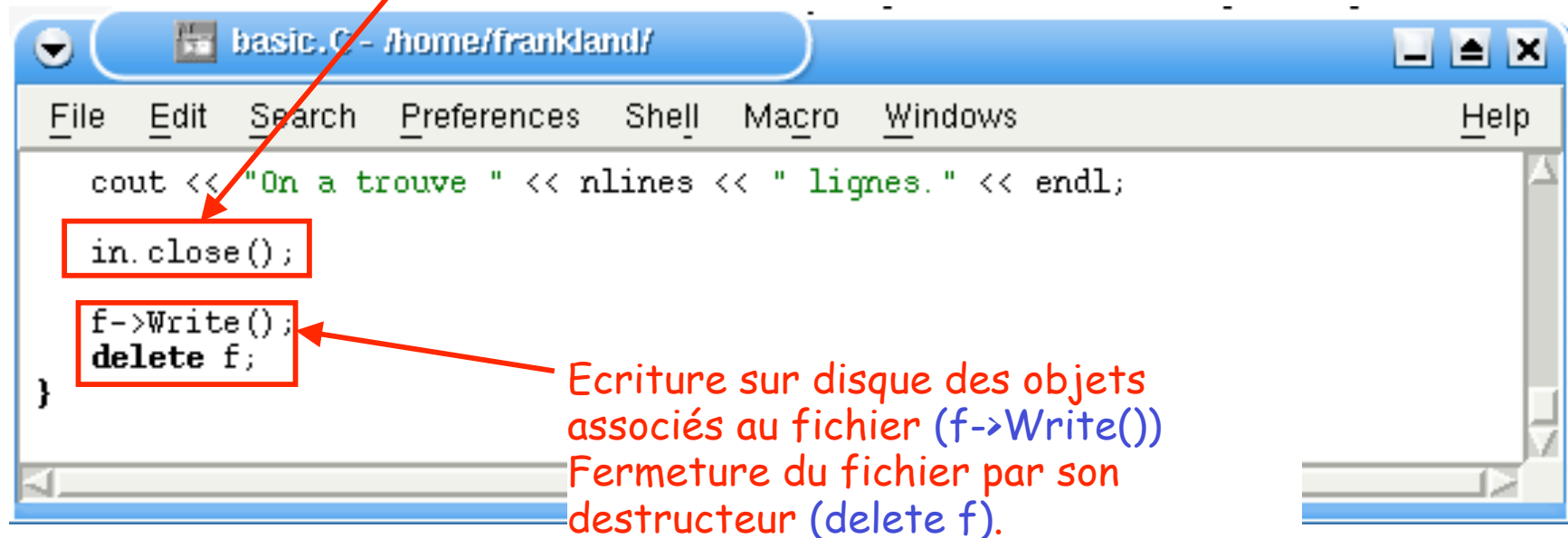
Affichage des valeurs lues à l'écran.
endl => aller à la ligne

Remplissage de l'histo et du ntuple

Incrémente le nombre de lignes lues

Exemple d'analyse

Fermeture du
fichier ASCII



```
basic.C - /home/frankland/  
File Edit Search Preferences Shell Macro Windows Help  
cout << "On a trouve " << nlines << " lignes." << endl;  
in.close();  
f->Write();  
delete f;  
}
```

Ecriture sur disque des objets
associés au fichier (f->Write())
Fermeture du fichier par son
destructeur (delete f).

WARNING:

les objets associés au fichier
n'existent plus en mémoire une fois
qu'il a été fermé.

Exécuter l'analyse et voir le résultat

- Compiler et exécuter:

```
.L basic.C+  
basic()
```

- Ouvrir le fichier et afficher le spectre "h1":

```
TFile* fich = new TFile("basic.root")  
fich->ls()  
TH1F* histo = (TH1F*)fich->Get("h1")  
histo->Draw()
```

Affiche le
contenu du
fichier

Les 2 dernières lignes en 1 seule:
fich->Get("h1")->Draw()

Copie en mémoire de
l'objet "h1".
Retourne l'adresse de
cette copie.

Dernières astuces

Des exemples de fonctions/scripts

- Sur le site web, rubrique [Tutorials](#), on trouve beaucoup d'exemples
- Attention! S'il n'y a pas d'en-tête avec déclaration d'une fonction, il faut faire:

Exécution
d'un script
sans en-tête

`.x toto.C`

Dans ce cas, le code est interprété au lieu d'être compilé.

WARNING: utilisation fortement déconseillée.

Ce que vous risquez en utilisant l'interpréteur...

L'interpréteur



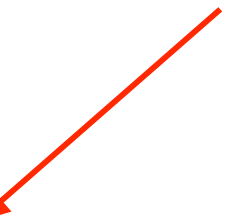
Vous

Le fichier rootlogon.C


- Ce fichier sans en-tête s'exécute automatiquement lorsqu'on lance ROOT dans le répertoire où il se trouve.

```
{  
gStyle->SetPalette(1);  
cout << "Salut " << gSystem->Getenv("USER") << "!" << endl;  
gSystem->Exec("date");  
}
```

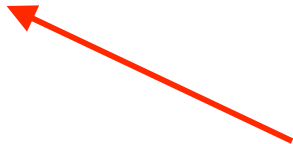
Pour avoir de
belles couleurs
dans les bidims



Exécute la
commande
système "date"



Retourne la valeur
de la variable
d'environnement
USER



Pour plus d'infos, voir les classes TStyle et TSystem

Exercice

Un autre exemple d'analyse de données

Exercice

(Episode 1: The ROOT menace)

http://caeinfo.in2p3.fr/root/Formation/fr/Jour2/exo_j2.data

- Faire un programme d'analyse du fichier **exo_j2.data** dans lequel on a 4 variable x,y,z et e (comme c'est original!)

**x de -25 à 25, y de -25 à 25,
z de -10 à 10, e de -500 à 2500**

- Générer les histogrammes suivants
 - Monodims: distribution de z (**TH1F**)
 - Bidims: y vs x, z vs x, z vs y (**TH2F**)
 - Profiles: <e> vs x, <e> vs y (**TProfile**)
 - Tridims: z vs y vs x, e vs y vs x (**TH3F**)
- Les sauvegarder dans le fichier **exo_j2.root**.

Exercice

(Episode 2: The return of *exo_j2.root*)

- Ouvrir **exo_j2.root**.
- Déterminer les 3 intervalles de z les plus peuplés.
 - *Notez-les!*
- En utilisant le FitPanel
 - fitter le TProfile <e> vs x avec un polynôme et *noter* les valeurs des 2 derniers paramètres *ex1* et *ex2*.
 - fitter le TProfile <e> vs y avec un polynôme et *noter* les valeurs des 3 derniers paramètres *ey1*, *ey2* et *ey3*.
- Fermer **exo_j2.root**.

Exercice

(Part 3: The analysis strikes back)

- Refaire un programme d'analyse du fichier **exo_j2.data**.
 - Générer les histogrammes suivants
 - Monodims: distribution (**TH1F**) de
$$de=e-ex1*x-ex2*x*x-ey1*y-ey2*y*y-ey3*y*y*y$$
 - Bidims: y vs x pour chaque intervalle en z déterminé dans la deuxième partie (**TH2F**)
 - Profiles 2D: $\langle z \rangle$ vs y vs x, $\langle e \rangle$ vs y vs x (**TProfile2D**)
 - Les ajouter au fichier **exo_j2.root**.

Exercice

(Part 4: The final shot)

- Déterminer la largeur de la distribution de **de** en la fittant avec une gaussienne.
- Ecrire un script permettant de visualiser dans le même TCanvas les quatre histogrammes suivants:
 - y vs x pour $z < 1$ avec l'option **col**
 - y vs x pour $3 < z < 5$ avec l'option **box**
 - $\langle z \rangle$ vs y vs x avec l'option **zcol**
 - $\langle e \rangle$ vs y vs x avec l'option **surf1**
- Sauvegarder l'image dans un fichier ".gif"