

ROOT tutorial

— part II —

1st April 2015

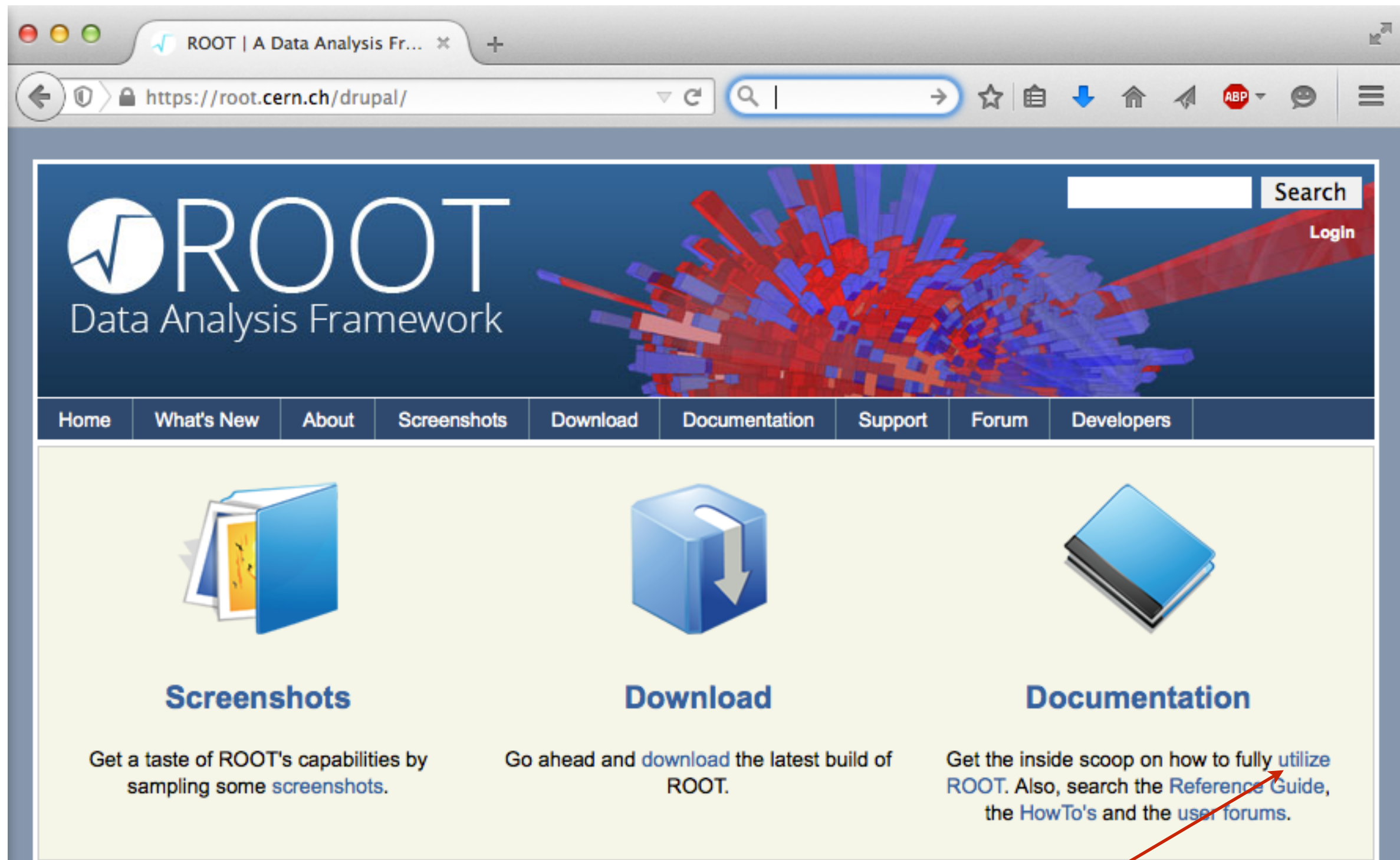
Adrian Perieanu



ROOT tutorial: goals

- * how to install it ✓
- * how to find/read documentation ✕
- * perform an interactive analysis with ROOT ✕
- * design and write own analysis macros ✕
- * how to store results of your analysis ✕

ROOT tutorial: how to find/read documentation



**we start from the same
point as before**

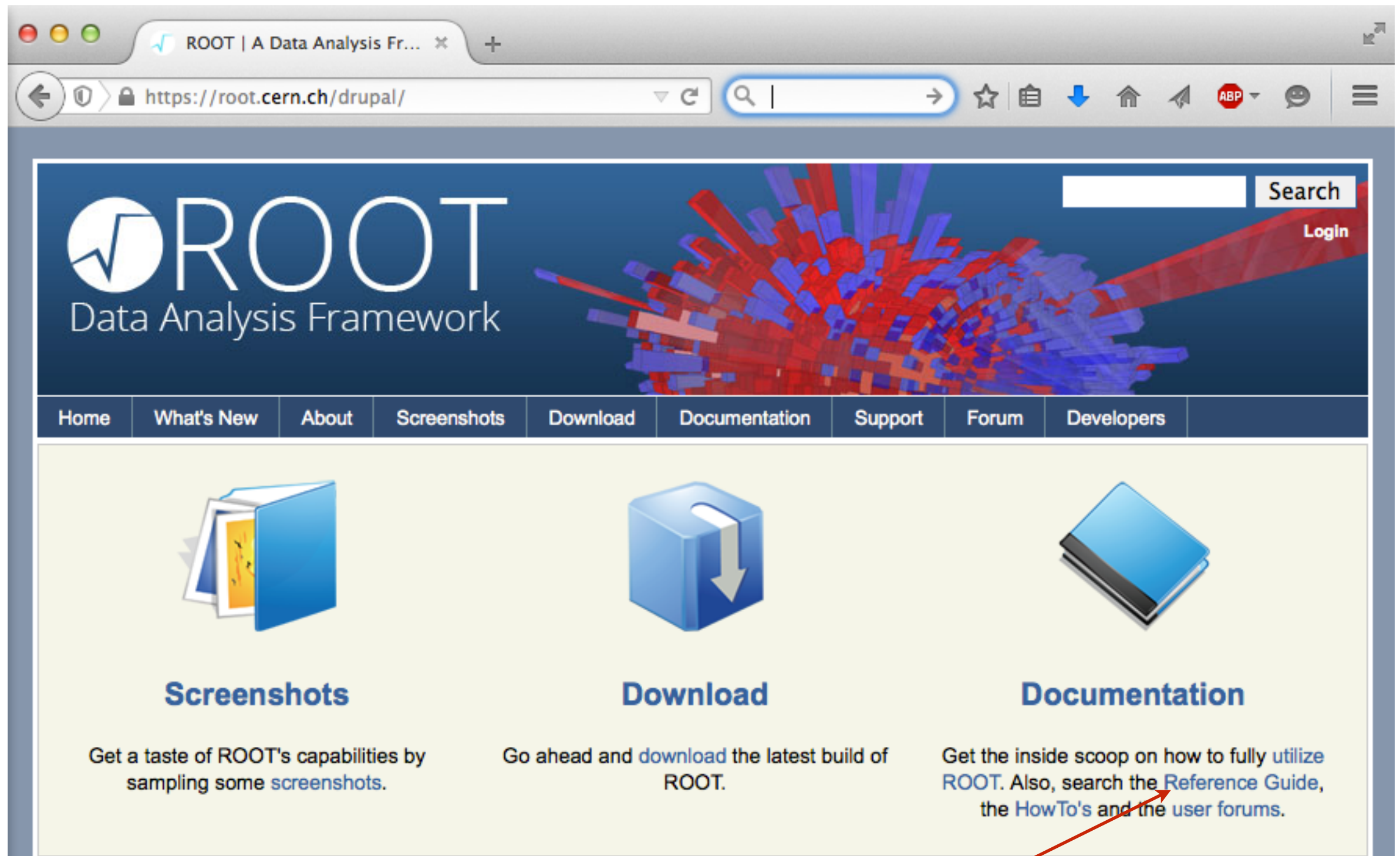
ROOT documentation: I. read

a series of manuals are waiting for you

The screenshot shows the ROOT Data Analysis Framework website. The browser address bar displays `https://root.cern.ch/drupal/content/users-guide`. The website header features the ROOT logo and a navigation menu with links: What's New, About, Screenshots, Download, Documentation, Support, Forum, and Developers. A search bar is located in the top right corner. The main content area is titled "User's Guide" and lists various manuals and guides available: ROOT User's Guide, ROOT Primer, TSpectrum manual, Minuit2 manual, HttpServer manual, RooFit Manual, and TMVA Manual. A red arrow points from the text "a series of manuals are waiting for you" to the "ROOT User's Guide" link. Below the list of manuals, there is a section titled "ROOT User's Guide" which states that the guide is written in Markdown format and allows for generating HTML, EPUB, PDF, etc. It also lists the development trunk (6.00.00), the pro version (5.34.00), and an old version of the complete User's Guide in PDF/A4 format for reference.

if you prefer not to read them, go to next page

ROOT tutorial: I'. don't read



go to "Reference Guide"

ROOT documentation:

II. choose your favourite version

The screenshot shows the ROOT Data Analysis Framework website. The browser address bar displays <https://root.cern.ch/drupal/content/reference-guide>. The website header features the ROOT logo and the text "Data Analysis Framework". A navigation menu includes links for Home, What's New, About, Screenshots, Download, Documentation, Support, Forum, and Developers. The "What's New" sidebar lists recent updates, including patch releases and development releases. The main content area, titled "Reference Guide", provides information about the documentation's availability for various ROOT versions. A list of versions is shown, with "Pro version 5.34.00" highlighted by a red box. A red arrow points from this box to a red callout box containing the text "my favoured version is still 5.34".

ROOT Data Analysis Framework

Home What's New About Screenshots Download Documentation Support Forum Developers

What's New

- March 24, 2015, 11:06
Patch release 5.34/28 - 2015-03-24
- February 20, 2015, 13:35
Patch release 5.34/26 - 2015-02-20
- February 16, 2015, 9:55
Patch release 6.02/05 - 2015-02-09
- January 28, 2015, 12:36
Development Release 6.03/02 - 2015-01-28

Home

Reference Guide

documentation


The Reference Guide is available for all major ROOT releases, and for the current development snapshot in subversion:

- **Pro version 6.00.00**
- **Pro version 5.34.00**
- Old version 5.32.00
- Old version 5.30/00
- Old version 5.28/00
- Old version 5.26/00
- Old version 5.24/00

my favoured version is still 5.34

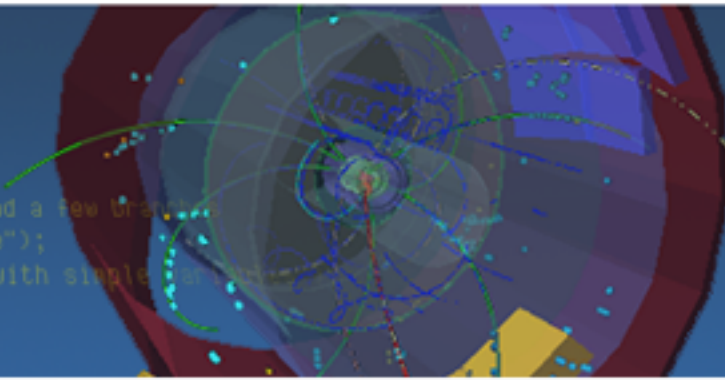
ROOT installation: II. choose

← <https://root.cern.ch/root/html534/ClassIndex.html> ☆



ROOT

```
//create the file, the Tree and a few branches
TFile f("tree1.root","recreate");
TTree t1("t1","a simple Tree with simple branches");
t1.Branch("px",&px,"px/F");
t1.Branch("py",&py,"py/F");
```



Quick Links: [ROOT Homepage](#) [Class Index](#) [Class Hierarchy](#)

ROOT

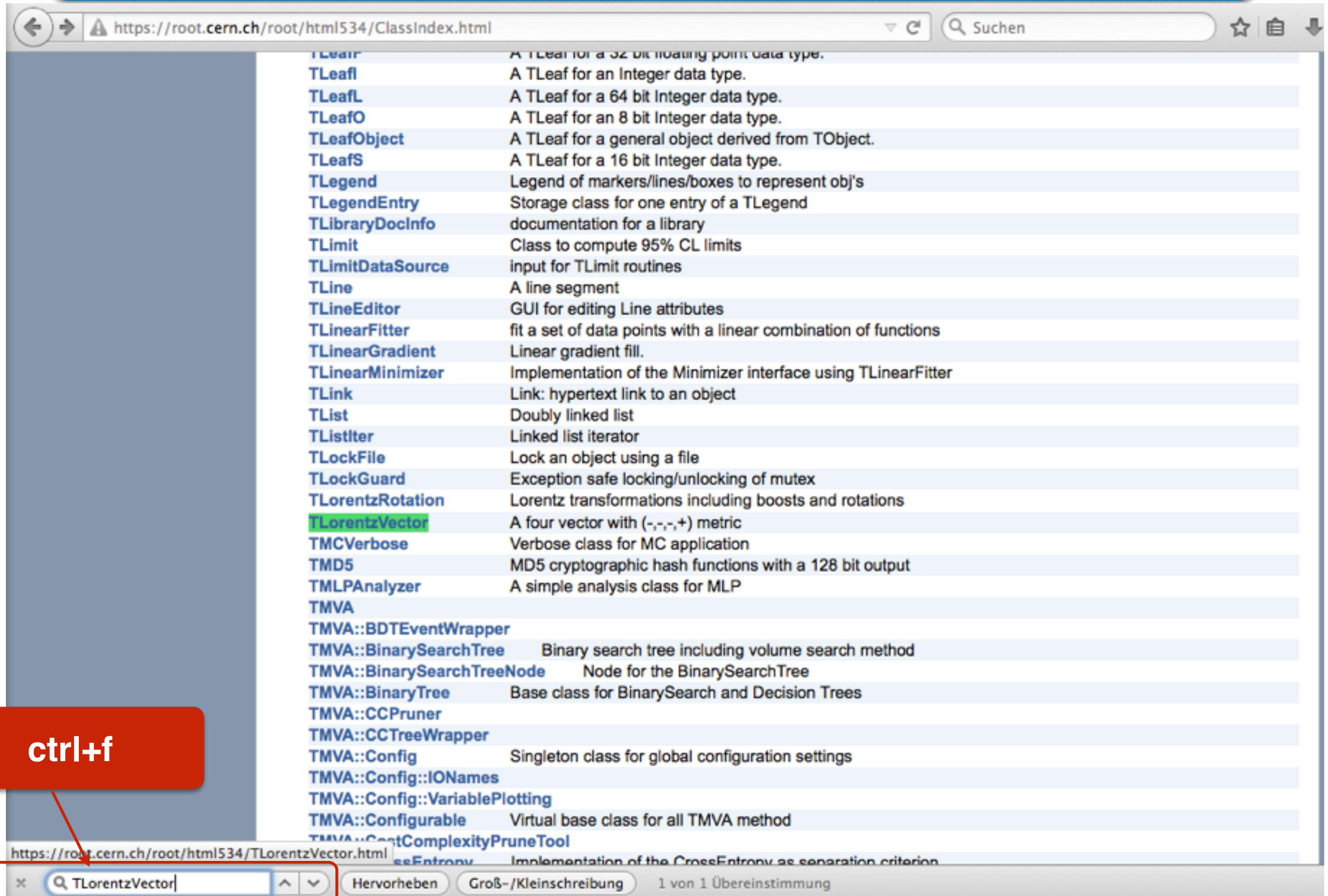
Class Index

Modules
[BINDINGS](#) [CINT](#) [CORE](#) [GEOM](#) [GRAF2D](#) [GRAF3D](#) [GUI](#) [HIST](#) [HTML](#) [IO](#) [MATH](#) [MISC](#) [MONTECARLO](#)
[NET](#) [PROOF](#) [ROOFIT](#) [SQL](#) [TEST](#) [TMVA](#) [TREE](#)

Jump to
[C](#) [ROOT:](#) [ROOT::Math:](#) [ROOT::Math::L](#) [ROOT::Math::P](#) [ROOT::Math::S](#) [ROOT::Math::SMatrix<f](#)
[ROOT::Math::T](#) [ROOT::T](#) [ROOT::TS](#) [Roo1](#) [RooC](#) [RooCh](#) [RooG](#) [RooN](#) [RooS](#) [RooSt](#) [RooStats:](#) [RooU](#)
[T](#) [TB](#) [TC](#) [TD](#) [TEv](#) [TEveG](#) [TEveQ](#) [TEveW](#) [TG](#) [TGH](#) [TGL](#) [TGLP](#) [TGLW](#) [TGR](#) [TGU](#) [TGeo](#)
[TGeoN](#) [TGeoT](#) [TGu](#) [TI](#) [TMV](#) [TMVA:](#) [TMVA::V](#) [TMe](#) [TO](#) [TP](#) [TPo](#) [TPy](#) [TR](#) [TS](#) [TSp](#) [TSu](#) [TU](#)
[TW](#)

[ColorStruct_t](#)
[CpuInfo_t](#) CPU load information.
[Event_t](#)
[FileStat_t](#)
[FontAttributes_t](#)
[FontMetrics_t](#)

ROOT installation: II. search



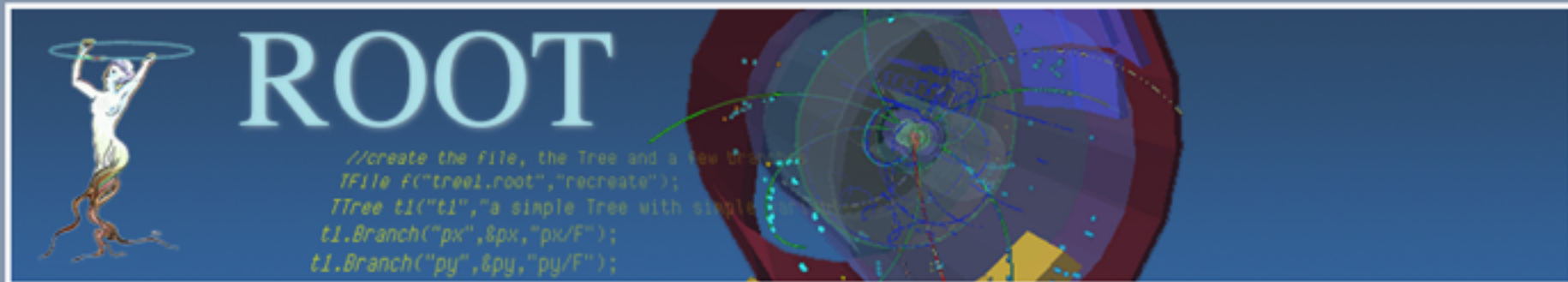
The screenshot shows the ROOT ClassIndex.html page in a web browser. The address bar displays the URL `https://root.cern.ch/root/html534/ClassIndex.html`. A search bar in the top right corner contains the text "Suchen". A list of classes is displayed on the right side of the page, including `TLorentzVector`, which is highlighted in green. A red box with the text "ctrl+f" and an arrow points to the search bar. The search bar contains the text "TLorentzVector". Below the search bar, the results show "1 von 1 Übereinstimmung".

Class	Description
<code>TLorentzVector</code>	A four vector with $(-, -, -, +)$ metric
<code>TMVA::BinarySearchTree</code>	Binary search tree including volume search method
<code>TMVA::BinarySearchTreeNode</code>	Node for the BinarySearchTree
<code>TMVA::BinaryTree</code>	Base class for BinarySearch and Decision Trees
<code>TMVA::CCPruner</code>	
<code>TMVA::CCTreeWrapper</code>	
<code>TMVA::Config</code>	Singleton class for global configuration settings
<code>TMVA::Config::IONames</code>	
<code>TMVA::Config::VariablePlotting</code>	
<code>TMVA::Configurable</code>	Virtual base class for all TMVA method
<code>TMVA::CostComplexityPruneTool</code>	
<code>TMVA::CrossEntropy</code>	Implementation of the CrossEntropy as separation criterion

ROOT installation: IV. inspect

https://root.cern.ch/root/html534/TLorentzVector.html

Suchen



```
//create the file, the Tree and a few branches
TFile f("tree1.root","recreate");
TTree t1("t1","a simple Tree with simple branches");
t1.Branch("px",&px,"px/F");
t1.Branch("py",&py,"py/F");
```

Quick Links:	ROOT Homepage	Class Index	Class Hierarchy	Search documentation...
Source:	header file	source file	viewVC header	Search
Sections:	class description	function members	viewVC source	class charts

ROOT » MATH » PHYSICS » TLorentzVector

class TLorentzVector: public TObject

The Physics Vector package

-* The Physics Vector package consists of five classes:

- * - [TVector2](#)
- * - [TVector3](#)
- * - [TRotation](#)
- * - [TLorentzVector](#)
- * - [TLorentzRotation](#)

-* It is a combination of CLHEP's Vector package written by
-* Leif Lonnblad, Andreas Nilsson and Evgueni Tcherniaev
-* and a ROOT package written by Pasha Murat.
-* for CLHEP see: <http://wwwinfo.cern.ch/asd/lhc++/clhep/>
-* Adaption to ROOT by Peter Malzacher
-*

TLorentzVector

TLorentzVector is a general four-vector class, which can be used either for the description of position and time (x,y,z,t) or

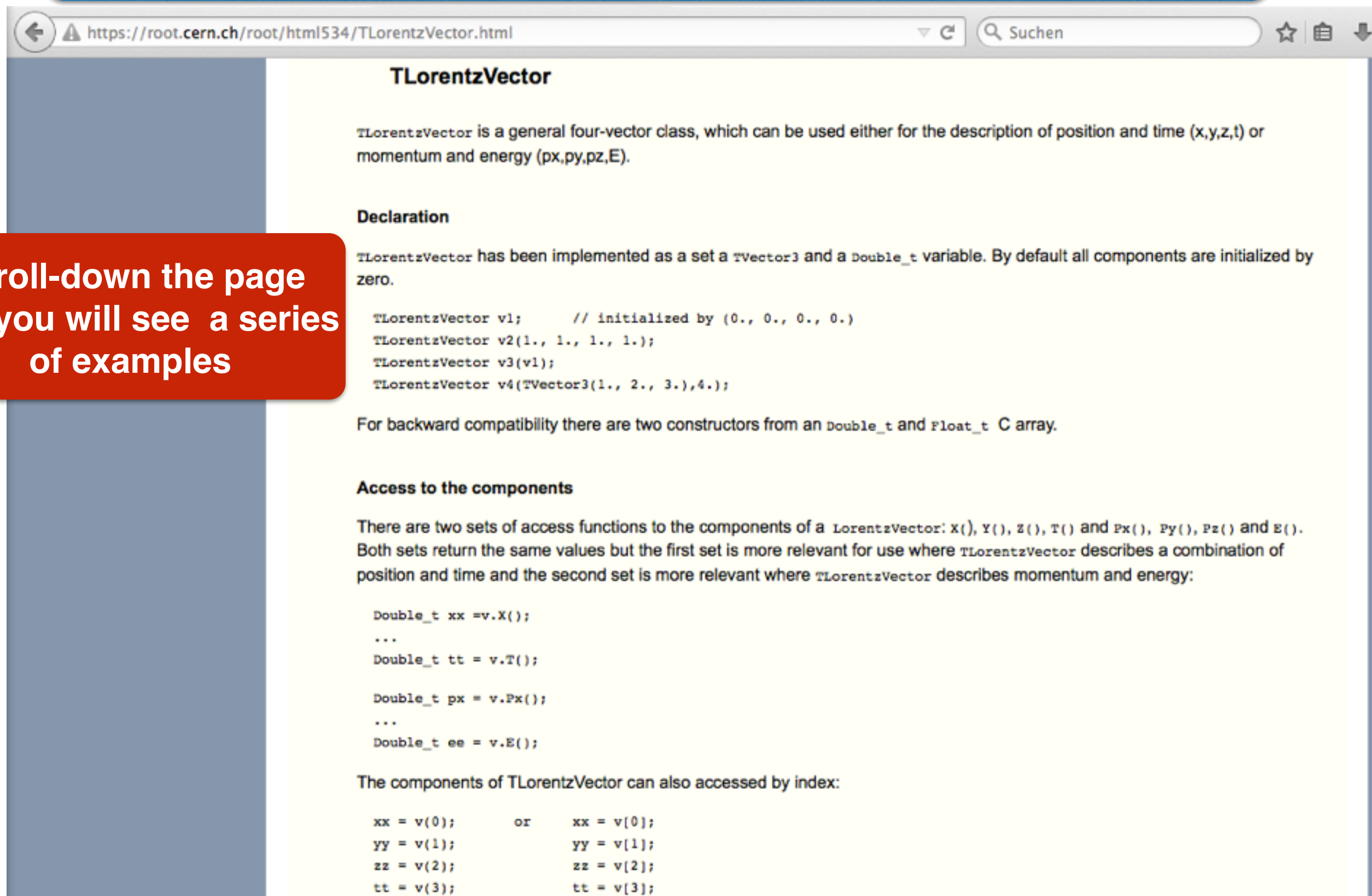
TLorentzVector

Hervorheben Groß-/Kleinschreibung 1 von 1 Übereinstimmung

to be able to access the object, it has to be defined as “public”

ROOT installation: IV. inspect some more

scroll-down the page
and you will see a series
of examples



The screenshot shows a web browser window with the URL `https://root.cern.ch/root/html534/TLorentzVector.html`. The page title is **TLorentzVector**. The content describes the `TLorentzVector` class as a general four-vector class for position and time or momentum and energy. It includes a **Declaration** section with a code example showing the initialization of four vectors: `v1` (initialized to zero), `v2` (1, 1, 1, 1), `v3` (copy of `v1`), and `v4` (constructed from a `TVector3` and a scalar). It also mentions backward compatibility constructors. The **Access to the components** section lists two sets of access functions: `X(), Y(), Z(), T()` and `Px(), Py(), Pz(), E()`. A code example shows how to use these functions to extract components into `Double_t` variables. Finally, it states that components can also be accessed by index, with a code example showing `v(0)` through `v(3)` and `v[0]` through `v[3]`.

TLorentzVector

TLorentzVector is a general four-vector class, which can be used either for the description of position and time (x,y,z,t) or momentum and energy (px,py,pz,E).

Declaration

TLorentzVector has been implemented as a set a TVector3 and a Double_t variable. By default all components are initialized by zero.

```
TLorentzVector v1;          // initialized by (0., 0., 0., 0.)
TLorentzVector v2(1., 1., 1., 1.);
TLorentzVector v3(v1);
TLorentzVector v4(TVector3(1., 2., 3.),4.);
```

For backward compatibility there are two constructors from an Double_t and Float_t C array.

Access to the components

There are two sets of access functions to the components of a LorentzVector: X(), Y(), Z(), T() and Px(), Py(), Pz() and E(). Both sets return the same values but the first set is more relevant for use where TLorentzVector describes a combination of position and time and the second set is more relevant where TLorentzVector describes momentum and energy:

```
Double_t xx = v.X();
...
Double_t tt = v.T();

Double_t px = v.Px();
...
Double_t ee = v.E();
```

The components of TLorentzVector can also accessed by index:

```
xx = v(0);      or      xx = v[0];
yy = v(1);      yy = v[1];
zz = v(2);      zz = v[2];
tt = v(3);      tt = v[3];
```

ROOT installation: IV. inspect some more

https://root.cern.ch/root/html534/TLorentzVector.html

Function Members (Methods)

public:

```
TLorentzVector ()
TLorentzVector (const Double_t* carray)
TLorentzVector (const Float_t* carray)
TLorentzVector (const TLorentzVector& lorentzvector)
TLorentzVector (const TVector3& vector3, Double_t t)
TLorentzVector (Double_t x, Double_t y, Double_t z, Double_t t)
virtual ~TLorentzVector ()
Double_t Angle (const TVector3& v) const
Double_t Beta () const
void Boost (const TVector3& b)
void Boost (Double_t, Double_t, Double_t)
TVector3 BoostVector () const
static TClass* Class ()
Double_t CosTheta () const
Double_t DeltaPhi (const TLorentzVector& v) const
Double_t DeltaR (const TLorentzVector& v) const
Double_t Dot (const TLorentzVector& q) const
Double_t DrEtaPhi (const TLorentzVector& v) const
Double_t E () const
Double_t Energy () const
Double_t Et () const
Double_t Et (const TVector3& v) const
Double_t Et2 () const
Double_t Et2 (const TVector3& v) const
Double_t Eta () const
TVector2 EtaPhiVector ()
Double_t Gamma () const
void GetXYZT (Double_t* carray) const
void GetXYZT (Float_t* carray) const
virtual TClass* IsA () const
Double_t M () const
```

scroll-down even more,
and you will find the
public methods

ROOT installation: let's try to use it

define 2 Lorentz Vectors

add the 2 Lorentz Vectors

extract the mass

```
perieanus-MacBook-Pro% root -l
root [0] TLorentzVector blume(1.,1.,1.,1.);
root [1] TLorentzVector baum(1.,1.,1.,2.);
root [2] TLorentzVector baum_hat_blume;
root [3] baum_hat_blume = baum+blume;
root [4] Double_t masse_blume = blume.M();
root [5] Double_t masse_baum_hat_blume = baum_hat_blume.M();
root [6] cout<<"masse_blume:"<<masse_blume<<" masse_baum_hat_blume:"<<masse_baum_hat_blume<<endl;
masse_blume:-1.41421 masse_baum_hat_blume:-1.73205
root [7] .q
```

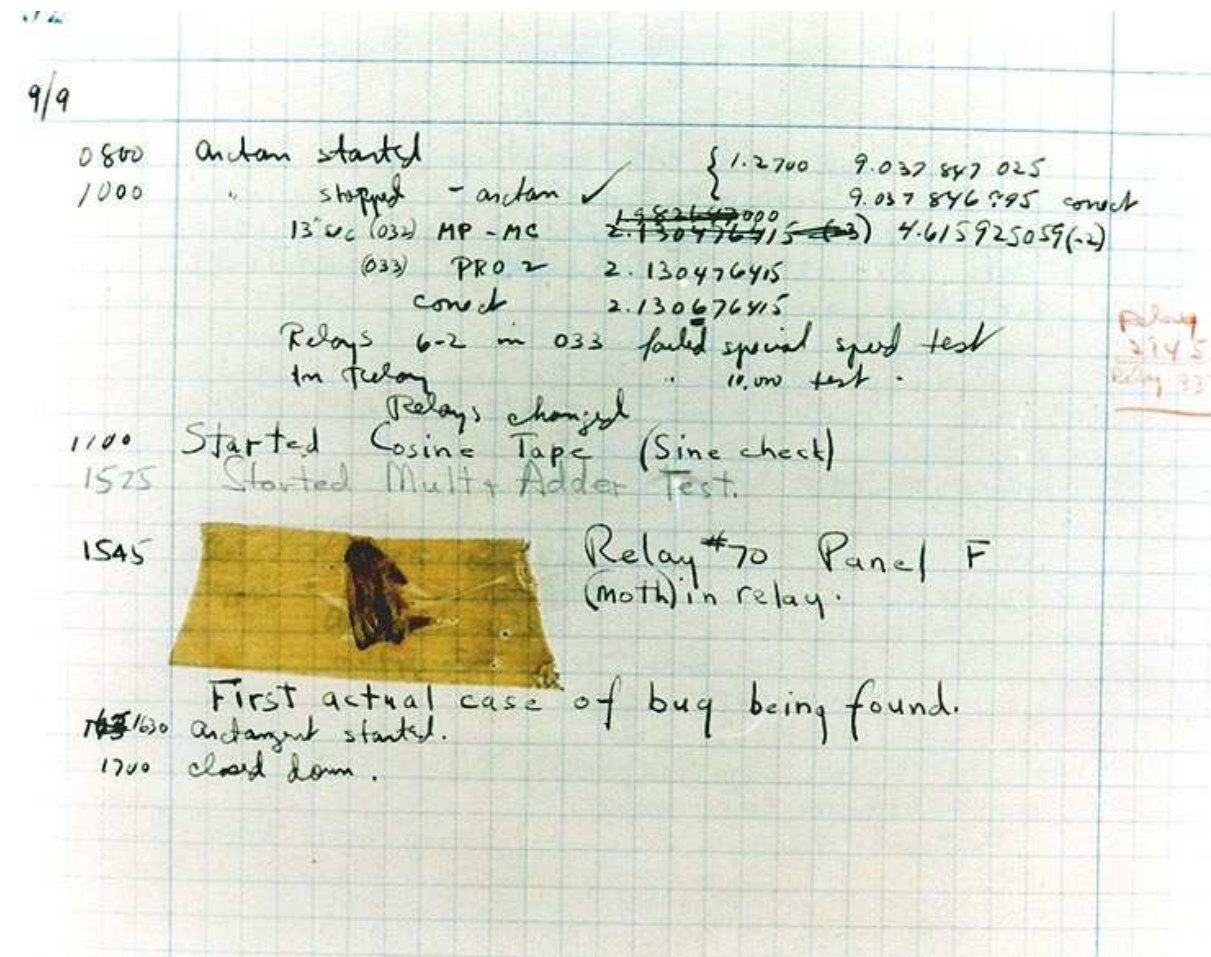
print out the mass

is the result correct?
how can you check it?
why is it important to check it?

every line of written code introduces potentially a (software) bug

what is a (software) “bug”?

what is a (software) “bug”?



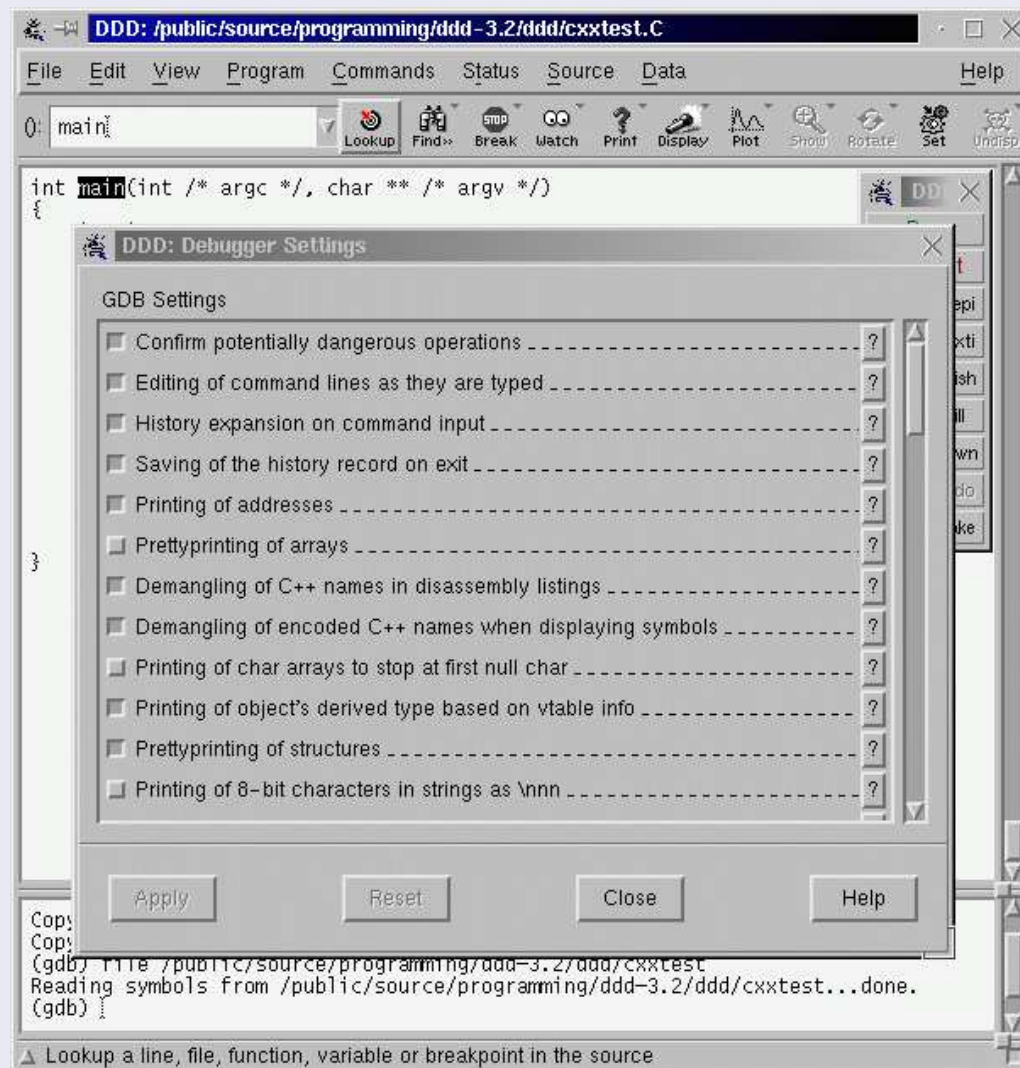
The first bug was really a bug: a moth which in 1947 went into a "big" computer at the Harvard University (more than 15 m long, 72 Byte memory, three additions per second) and made a shortcut

Someone took the bug and "posted" in the logbook (a very special way of Debugging...)

there are also debugging tools

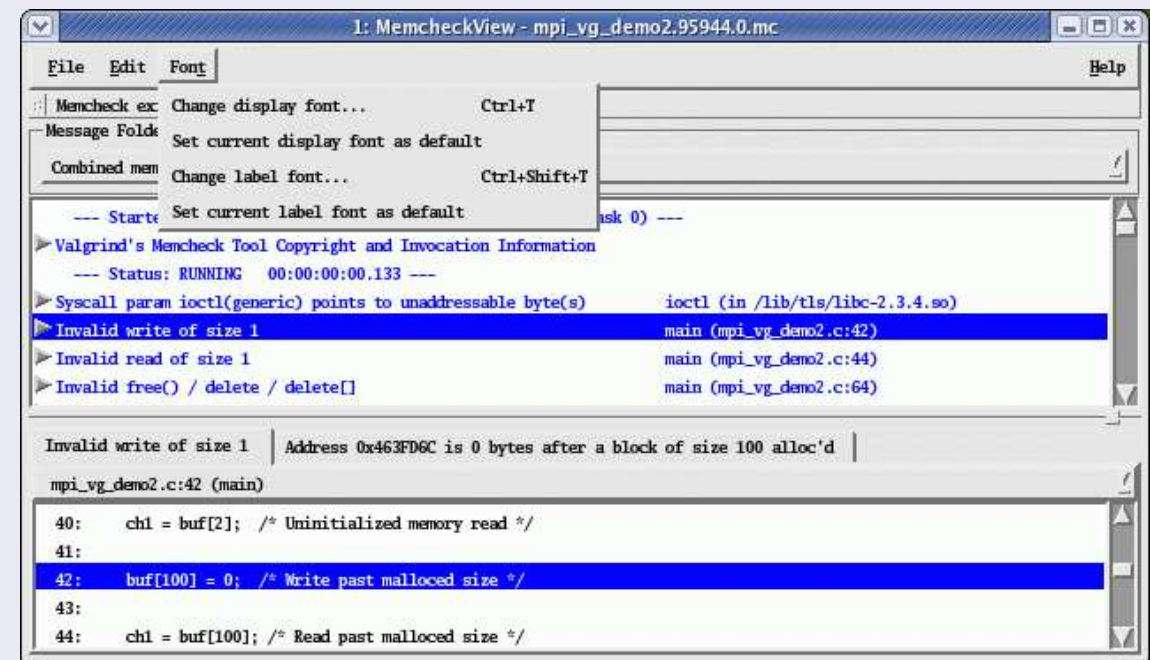
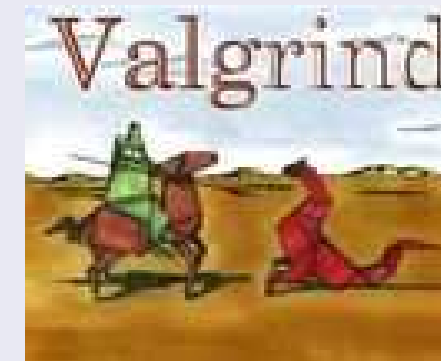
gdb

- gdb - der GNU-Debugger, a Unix tool, with its user interface, ddd



Valgrind

- valgrind for x86-Linux Programs



**the best way is when one understands its
own bugs!**

bugs will become your day by day companions

some of the famous bugs:



- 26.09.1983: A Soviet Satellite indicated that 5 ballistic missiles were launched from US.
- Possible Consequence: the World War III.
- But: the Soviet duty officer reasoned that if the U.S. was really attacking, more than five missiles would have been launched.
- A bug in the Soviet software failed to filter alarms caused by sunlight reflecting off cloud-tops.

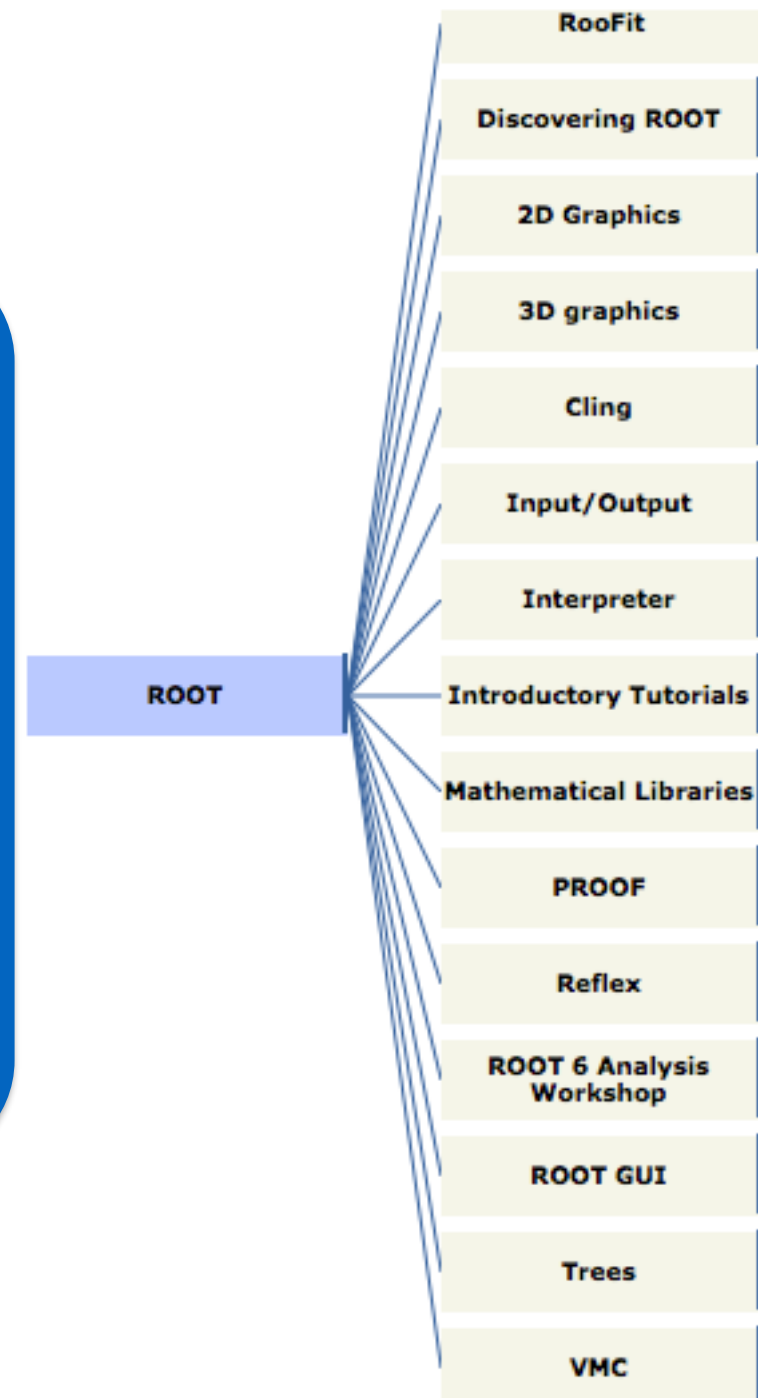
- Sometime the bugs have also "good" sides



- 2001 at the Open-Air-Festival in Sziget (Ungarn): an ATM gave double the amount of the requested money, but subtracted from the account the requested amount (as written on receipt...)

summary: documentation

- * go to: <https://root.cern.ch/drupal/content/documentation>
- * if inpatient:
<https://root.cern.ch/drupal/content/reference-guide>
- * choose your favoured version and search for the class you need
- * once you have implemented your code, check it for bugs
(you better find them before you start writing your thesis 😊 😊 😊)



time for a break and (a lot of) questions

almost half way through, from now on we can play with data