# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak


- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*

# Giriş

## ROOT nedir ?

- **Bir kütüphane**: Özelleşmiş daha küçük kütüphanelerden oluşan **büyük bir class kütüphanesidir**. ROOT kütüphaneleri, geliştirilen bağımsız uygulamalarda herhangi bir kütüphaneymiş gibi kullanılabilir. Örnegin:
  - **Kullanıcı arayüzü** (GUI) geliştirme kütüphanesi olarak
  - **Signal-slot** mekanizmasını sunan bir kütüphane olarak
  - **Veri analizi** için gerekli işlevleri sağlayan bir kütüphane olarak

- **Bir C/C++ yorumlayıcısı**: ROOT, CINT (C interpreter) adlı bir C yorumlayıcısı ile birlikte gelir. Bu yorumlayıcı, **hızlı algoritma geliştirme** için kullanışlıdır çünkü derleme işlemine gereksinim olmaksızın program çalıştırmaya imkan verir. CINT, C ve C++ komutlarını **sanki kabukta** çalıştırıyormus gibi tek tek girmeyi veya bir **kütükten okumayı** mümkün kılar
  - Bu, **görece yavaş çalışan** programlar üretir, fakat:
  - **Uygulama geliştirme** süreci daha hızlıdır; üstelik gerekli olması durumunda bu betikler, ***.so** nesne veya kütüphanelerine dönüştürülebilir ve daha hızlı çalıştırılabilir

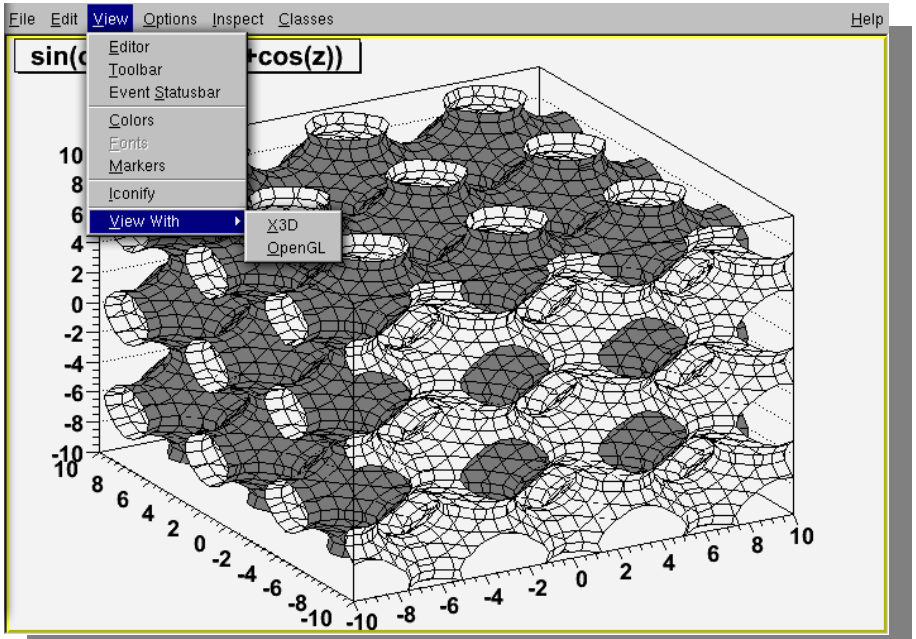
- **Bir çalışma ortamıdır**: ROOT fizikçilerin rahat çalışabilmeleri için bir çalışma ortamı sağlar
  - Yorumlayıcı, histogram ve analiz işlevleri, GUI geliştirme, I/O işlevleri, class kütüphanesi, paralel ve soket programlama, ağ haberleşmesi, v.b. kütüphaneleri **kullanıma hazır** şekilde sunar ve **uygulama geliştirmeyi hızlandırır**

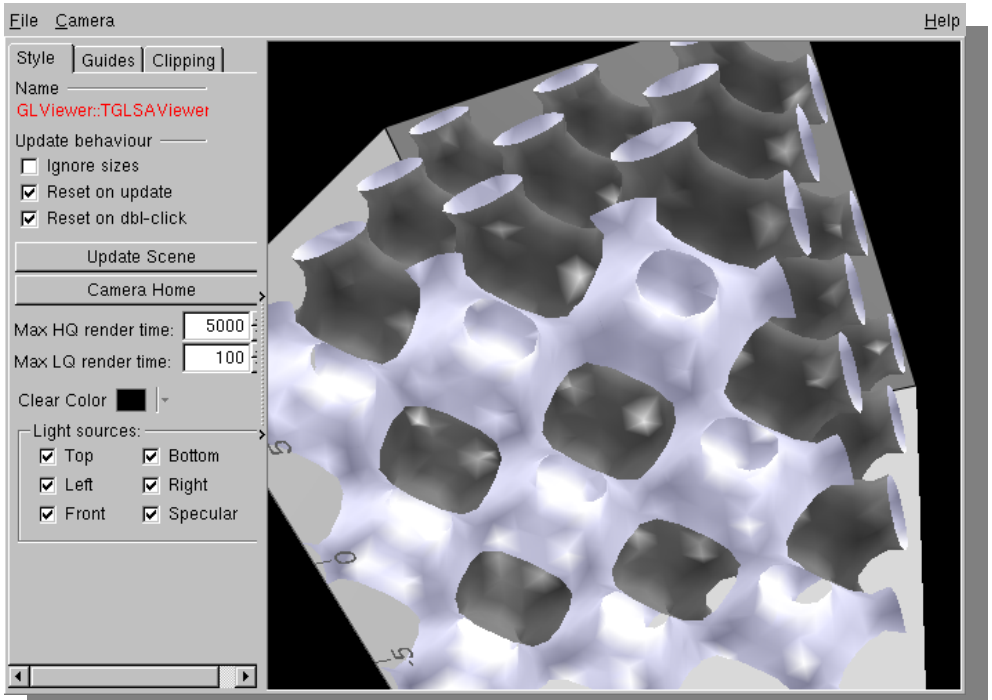
# Giriş

## ROOT nedir ?

- Çalışma ortamında (framework) olmanın **getirileri**:
  - Belirli bir işlevsellik için çok **daha az kaynak** yazmanın gerekliliği
  - Sonuçta geliştirilen programın **yüksek güvenilirliğe** sahip olması
  - Kendi içinde daha **tutarlı** bir *class* birlikteliği
  - Parçalanabilir, dolayısı ile **yeniden kullanılabilir** esnek mimari
  - Geliştirici **kendi alanına** daha fazla yoğunlaşabilir

```
root [0] TF3 f1("MerabaDunya","sin(cos(x)+sin(y)+cos(z))",-10,10,-10,10,-10,10)
root [1] f1->Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2] MerabaDunya->SetTitle("Bashka bi shey !..")
```

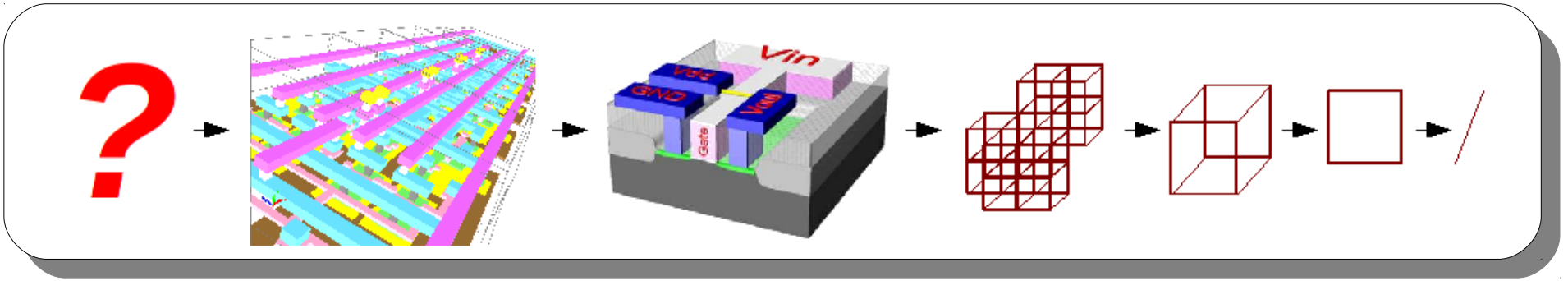


# Giriş

## Neden iyidir ?

- **Çalışma ortamının** (framework) **nesne yönelimli** (object oriented) olmasının getirileri:
  - Nesne yönelimli programlamaya özgü olarak verinin nesne içine gömülü (encapsulation) oluşu, verinin daha **soyut** düzeylerde ifadesini (abstraction) kolaylaştırdığı için, kitaplığı oluşturan parçaların (class) **yeniden kullanılırlığını** artırır
  - Class' tan class **türetme**, varolan nesneleri geliştirmeyi ve değiştirmeyi olanaklı kılar
  - Class' lar arasındaki **ast-üst ilişkisi** (hiyerarşi), **gerçek dünyadaki nesnelerin** daha kolay oluşturulmasına imkan verir.




  - Geliştirilen **yazılımın karmaşıklığı** düşük seviyede kalır, bilgi class' lar içinde düzenli biçimde yerleşiktir; tekdüze bir kaynak içinde küresel değişkenler (globals) üzerinde dağılmış durumda değildir.
  - Yazılımdan nesnelerin çıkarılması ya da yeni nesnelerin eklenmesi, **genel mimarii değiştirmez** ve böylece belirli bir mimari tekrar tekrar kullanılabilir

# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak


- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*

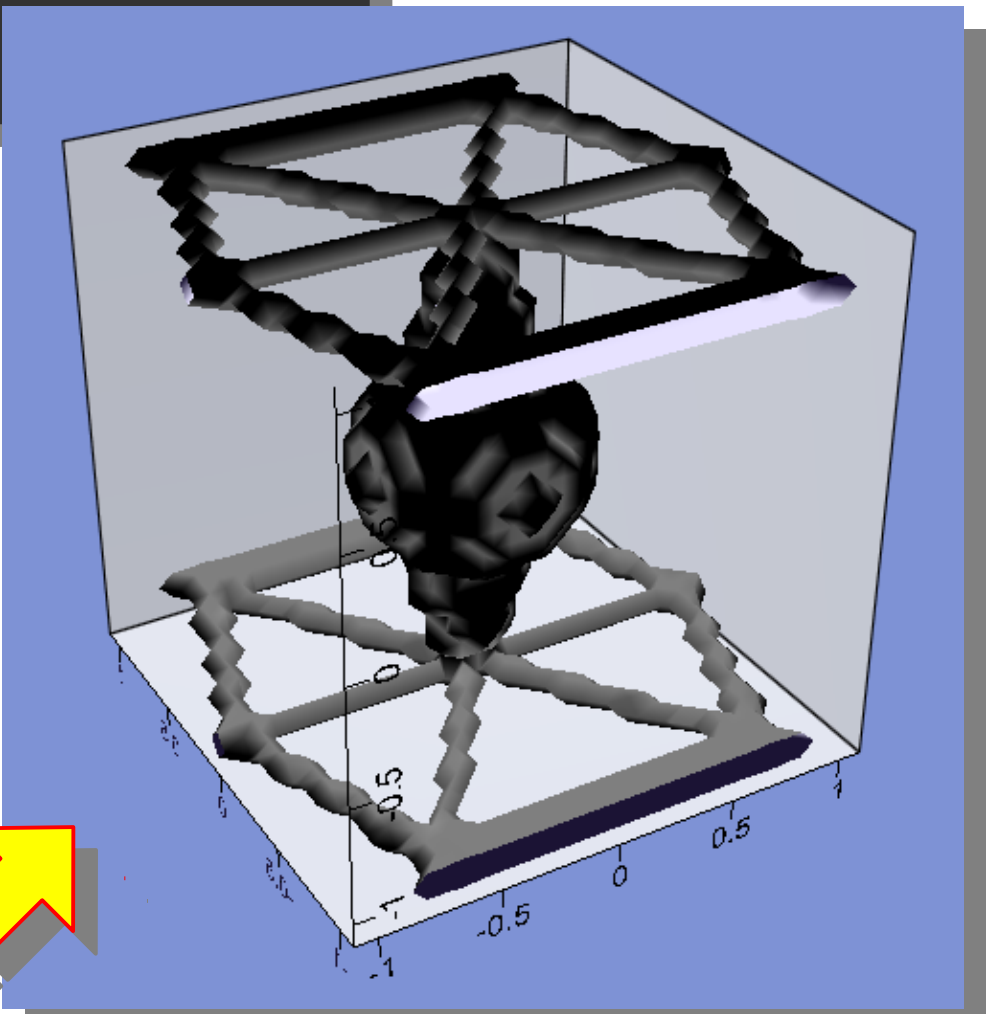
# Yükleme ve *Tie Fighter* Matematiği

## İlk etkileşim

```
> tar xvfz root_v5.22.00.source.tar.gz
> export ROOTSYS=$HOME/root
> export PATH=$PATH:$ROOTSYS/bin
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
> cd root
> ./configure
> make
> sudo make install
> _
```

- Çevresel **degişkenleri** ayarla, **derle** ve ROOT kullanıma **hazır**

```
# root -l
> TF3 *tieFighter = new
  TF3("tieFighter","(x^2+y^2+z^2<0.2)+
  ((y^2+z^2<0.08)*(x<0.4)*(x>0))+
  (x^2+4*y^2<(1-TMath::Abs(z))*0.12)+
  ((TMath::Abs(z)<0.95)*(TMath::Abs(z)
  >0.9)*(TMath::Abs(x)
  +TMath::Abs(y)*0.3<1))+
  ((TMath::Abs(z)<1)*(TMath::Abs(z)>0.
  89))*((TMath::Abs(x)<0.7)*(TMath::Ab
  s(y)>0.9)+(TMath::Abs(y)<0.035)+
  (x>y*0.7-0.05)*(x<y*0.7+0.05)+(-
  x>y*0.7-0.05)*(-x<y*0.7+0.05)+
  ((TMath::Abs(x)
  +TMath::Abs(y)*0.3<1.05)*(TMath::Abs
  (x)+TMath::Abs(y)*0.3>0.95)))",-
  1.1,1.1,-1.1,1.1,-1.1,1.1);
> tieFighter->Draw()
```

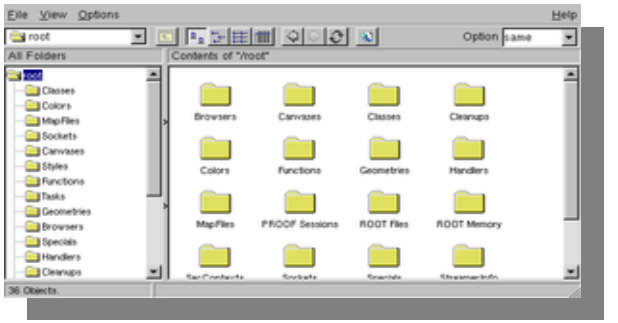


# Kullanım

## Komut vermek

- Kabukta çalışır gibi komut girmek
  - C/C++ ifadeleri ve tüm ROOT class kitaplığı, "#include<stdio.h>" ya da "int main()" gibi her programda bulunan ifadelere **gerek kalmadan** girilebilir. Örneğin aşağıdaki komut ile, bir TBrowser nesnesi yaratıyor ve adına BenimGuzelGozAticim diyoruz:

```
root [0] TBrowser BenimGuzelGozAticim
root [1] _
```



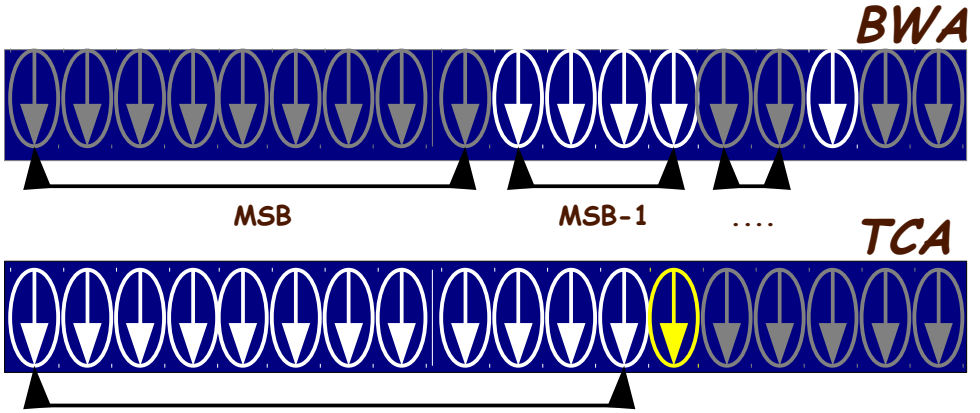

  - Ya da bir döngü:

```
root [0] for (int i=0 ; i<10 ; i++) {
end with '}', '@':abort > printf("%d nin karesi %d \n", i, i*i);
end with '}', '@':abort > }
0 nin karesi 0
1 nin karesi 1
2 nin karesi 4
3 nin karesi 9
4 nin karesi 16
5 nin karesi 25
6 nin karesi 36
7 nin karesi 49
8 nin karesi 64
9 nin karesi 81
root [1] _
```

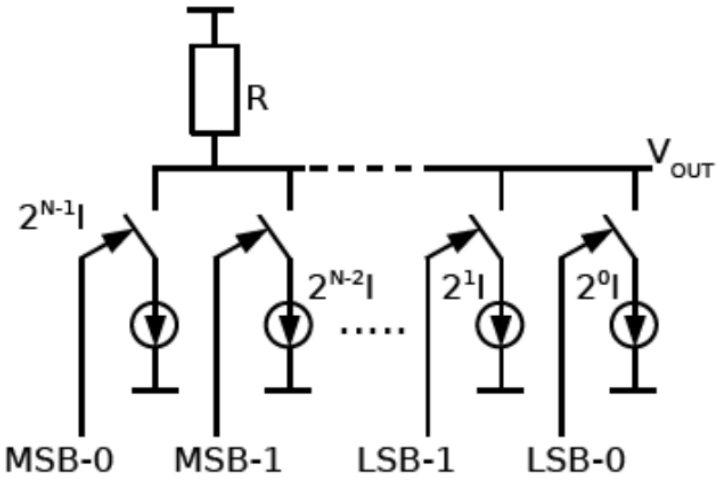
# Kullanım

## Betik yazmak (Mimari Karşılaştırıcı)

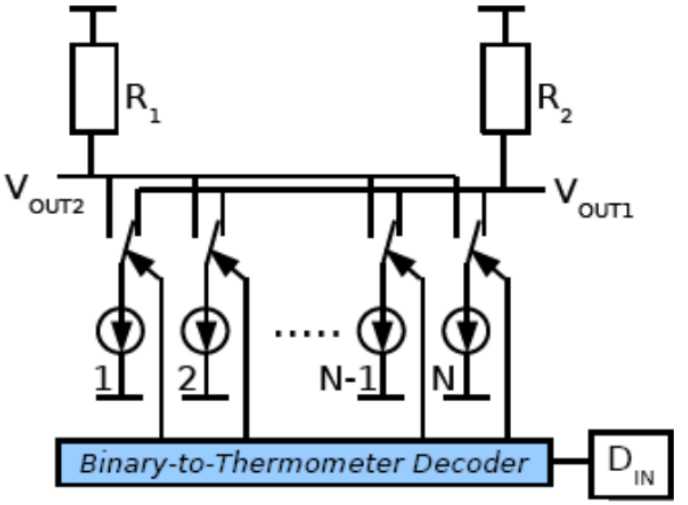
- 10-Bit **D/A çevirici** tasarlanacak
- İki **farklı mimari** var
- **Hangisi** daha uygun belirlemeliyiz ?
- **Ölçülebilir karşılaştırma** gerekli
- **Monte Carlo** (MC) yapılmalı




- Çan eğrisinden, $x_c=1$ ve $\sigma=0.02$ olmak üzere **rastlantısal sayı** üret
- Bunlar her iki D/A'yı da oluşturacak birim akım kaynakları olsun:
  - **BWA Durumunda:** İlk $2^{(N-1)}$ tanesinin toplamı MSB olsun ve **bir sonraki** $2^{(N-2)}$ tanesinin toplamı MSB'den sonraki bit ve **bir sonraki** $2^{(N-3)}$ tanesinin toplamı MSB'den iki sonraki ...
  - **TCA Durumunda:** **girişteki her adım tek bir birim akım kaynağının** daha çıkışa yönlendirilmesi olarak değerlendirilsin
- **MC** ile her iki mimari için de toplam (integral non-linearity, INL) ve adım başına (differential non-linearity, DNL) doğrusallıktan sapma miktarlarını ve bunların RMS' lerini hesapla




*İkili Ağırlıklı (BWA)*




*Termometre-Kodlu (TCA)*

# Kullanım

## Betik yazmak (Mimari Karşılaştırıcı)

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici.C | mimariKarsilastirici2.C

```
27 {
28    gROOT->Reset();
29
30    // edit these parameters for other simulations -----------
31    int noofbits=10;
32    int noofiteration4rms=100, nob=100; // nob : no of bins in distro
33    double sigma=0.02, centroid=1.0;
34    const Int_t kUPDATE = 50;
35    bool update=kTRUE;
36    //--------------------------------------------------------
37
38    int nop=pow(2.0, noofbits);
39    int boyut = pow(2, 1*noofbits);
40    double Iu[boyut]; // 2^noofbits current source
41    int lastincremented=0;
42    TCanvas *c1 = new TCanvas("c1","Binary vs Thermometer",1200,850);
43    TPad *pad1 = new TPad("pad1","Unit Current Source",0.0,0.0,1.0,1.0);
44    pad1->Draw();
45    pad1->Divide(5, 4, 0.001, 0.001);
46    pad1->cd(5)->SetFillColor(47);  pad1->cd(5)->SetGrid() ;
47    pad1->cd(6)->SetFillColor(47);  pad1->cd(6)->SetGrid() ;
48    pad1->cd(7)->SetFillColor(47);  pad1->cd(7)->SetGrid() ;
49    pad1->cd(8)->SetFillColor(47);  pad1->cd(8)->SetGrid() ;
50    pad1->cd(9)->SetFillColor(47);  pad1->cd(9)->SetGrid() ;
51    pad1->cd(10)->SetFillColor(47); pad1->cd(10)->SetGrid();
52    pad1->cd(11)->SetFillColor(21); pad1->cd(11)->SetGrid();
53    pad1->cd(12)->SetFillColor(21); pad1->cd(12)->SetGrid();
54    pad1->cd(13)->SetFillColor(21); pad1->cd(13)->SetGrid();
55    pad1->cd(14)->SetFillColor(21); pad1->cd(14)->SetGrid();
56    pad1->cd(15)->SetFillColor(21); pad1->cd(15)->SetGrid();
57    pad1->cd(16)->SetFillColor(21); pad1->cd(16)->SetGrid();
58
59    pad1->cd(1);
60    TH1F *h1f = new TH1F("h1f", "Test random numbers",nob,0.9,1.1);
61    TH1F *h1f_ins = new TH1F("h1f_ins", "Current Iu[i] Set", nob, 0.9, 1.1);
62    TH1F *h1f2= new TH1F("h1f2","Test random numbers COPY",nob,0.9,1.1);
63    TGraph *graph = new TGraph(nop);
64    graph->SetMarkerSize(0);
65    TF1 *gaus = new TF1("gaus","gaus(0)",0.9,1.1);
66    gaus->SetParameters(1, centroid, sigma);
67    gaus->Draw();
68    if (update) c1->Update();
69
70    TH1F *rms_dnl = new TH1F("rms_dnl", "DNL in RMS [LSB unit]",nop-2, 0, nop-2);
71    TH1F *rms_inl = new TH1F("rms_inl", "INL in RMS [LSB unit]",nop, 0, nop);
72    TH1F *rms_dac = new TH1F("rms_dac", "D/A Output in RMS [LSB unit]",nop, 0, nop);
73    TH1F *b_rms_dnl = new TH1F("b_rms_dnl", "b DNL in RMS [LSB unit]",nop-2, 0, nop-2);
```

Ln 216, Col 21 INS

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici.C | mimariKarsilastirici2.C

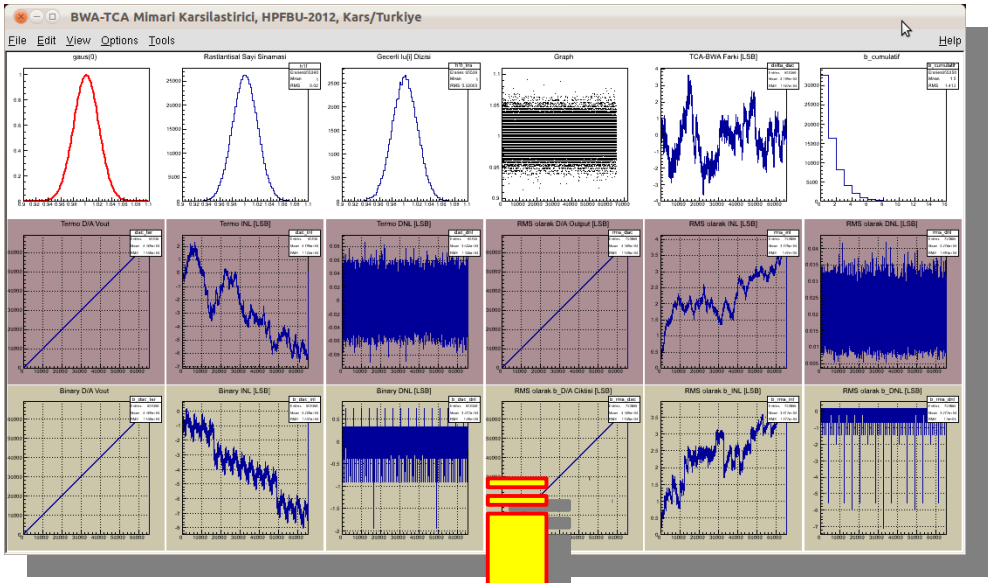
```
182         b_inl[i]=(i+1.0-b_dac[i]);
183         b_rms_inl->SetBinContent(i, b_rms_inl->GetBinContent(i)+b_inl[i]*b_inl[i]);
184         if (i>0 && i<nop) {
185             b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
186             b_rms_dnl->SetBinContent(i-1, b_rms_dnl->GetBinContent(i-1)+(b_dnl[i-1]-Iu[0
187         }
188         b_dac_inl->SetBinContent(i, b_inl[i]-Iu[0]);
189         if (i>0) b_dac_dnl->SetBinContent(i-1, b_dnl[i-1]);
190     }
191     pad1->cd(12);
192     b_dac_inl->Draw();
193     pad1->cd(13);
194     b_dac_dnl->Draw();
195     if (update) c1->Update();
196     pad1->cd(17);
197     delta_dac->Draw();
198     pad1->cd(18);
199     b_cumulatif->Draw();
200     pad1->cd(11);
201     b_dac_ter->Draw();
202
203   } // end loop
204
205     for (int i=0 ; i<nop ; i++) { // calculate RMS
206         rms_dac->SetBinContent(i, sqrt(rms_dac->GetBinContent(i)/noofiteration4rms));
207         if (i<nop-1) rms_dnl->SetBinContent(i, sqrt(rms_dnl->GetBinContent(i)/noofiterat
208         rms_inl->SetBinContent(i, sqrt(rms_inl->GetBinContent(i)/noofiteration4rms));
209         b_rms_dac->SetBinContent(i, sqrt(b_rms_dac->GetBinContent(i)/noofiteration4rms))
210         if (i<nop-1) b_rms_dnl->SetBinContent(i, 1.55-3*(-Iu[0]/2+sqrt(b_rms_dnl->GetBin
211         b_rms_inl->SetBinContent(i, sqrt(b_rms_inl->GetBinContent(i)/noofiteration4rms))
212     }
213     pad1->cd(8);
214     rms_dac->Draw();
215     pad1->cd(9);
216     rms_inl->Draw();
217     pad1->cd(10);
218     rms_dnl->Draw();
219     pad1->cd(14);
220     b_rms_dac->Draw();
221     pad1->cd(15);
222     b_rms_inl->Draw();
223     pad1->cd(16);
224     b_rms_dnl->Draw();
225     if (update) c1->Update();
226
227   gBenchmark->Stop("binary_vs_thermometer");
228 }
```

Ln 216, Col 21 INS

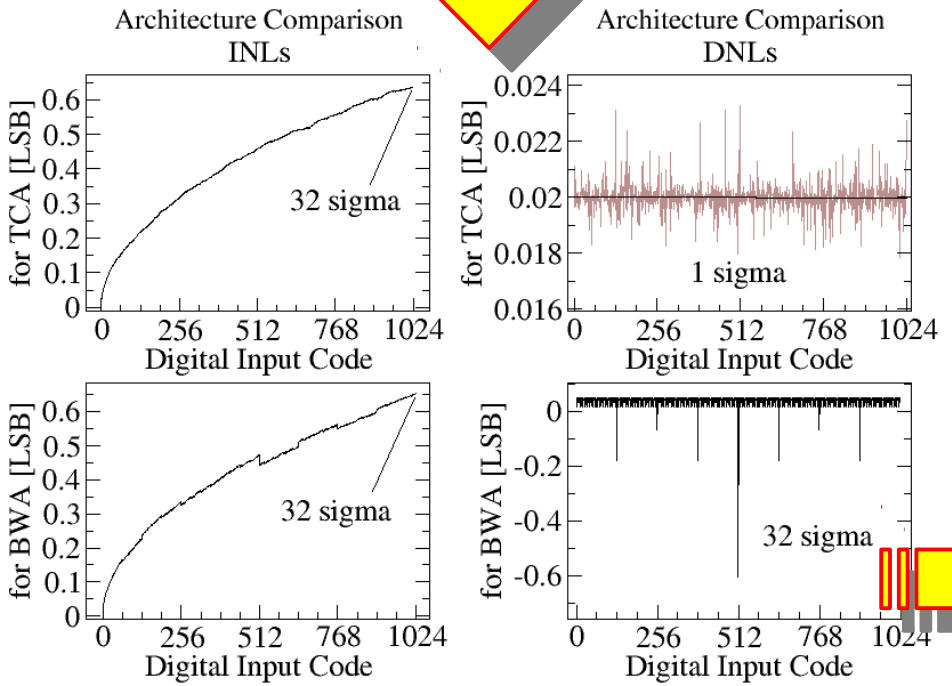
- Betiğin **isim taşımıyor** olduğuna dikkat edin
- Nasıl başlayıp nasıl bittiğine dikkat edin (**header** ve **return yok**)

# Kullanım

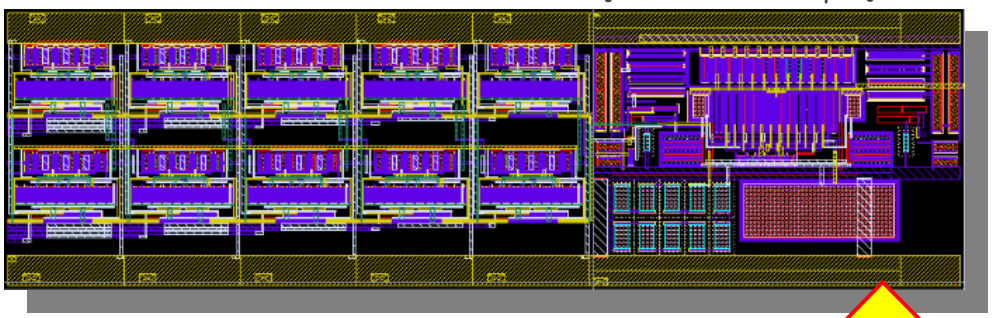
## Betik yazmak (Mimari Karşılaştırıcı)

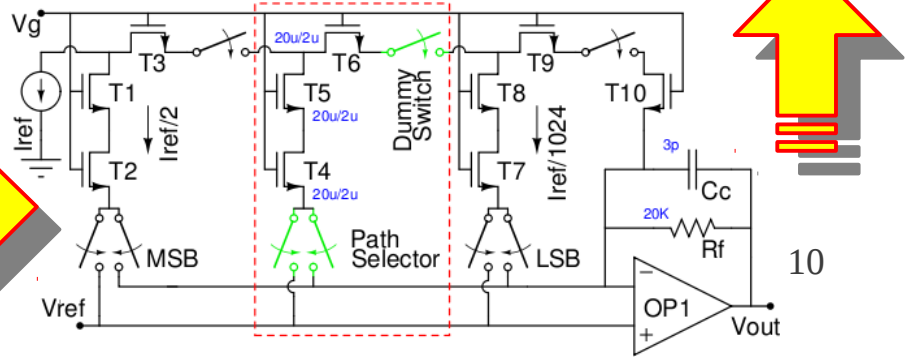



- **Seçimimi** yaptım:
  - **INL**' ler çok farklı değil
  - **DNL**' ler farklı, TCA' nin davranışı çok **daha iyi**, ama **çok yer** kaplayacak
- D/A' ımı **tasarladım** ve dedektör çalışanlara **teslim ettim**
- Çalıştayda **sundum**
  - **Görev** tamam !!




Halen CERN'deki COMPASS deneyinin RICH-I algıcını okumakta olan CMAD tümleşik devresinin bir parçasıdır.





# Kullanım

## Betiği *.so kitaplığına derlemek

- Mimari karşılaştırıcım betik halindeyken CINT tarafından **yorumlandığı** için yeterince **hızlı** sonuç **vermeyebilir**
- Onu **derlediğimde** ise yorumlanmak yerine, **çalışacaktır** ve bu çok **daha hızlı** bir süreçtir
- Betiğimde **hemen hiç değişiklik yapmadan**, onu bir ***.so** kitaplığına derleyebilirim (tıpkı cizgi ve ok class' larında olduğu gibi)
- Hemen hiç derken:
  - Kullanılan class' ların, başlık (header) kütüklerini eklemeliyim (TCanvas kullanılmış ise "**#include<TCanvas.h>**" gibi)
  - Yazdığım işleve (fonksiyon) tercihen betiğin saklandığı kütükle aynı olan bir isim vermeliyim (**isim.C** kütügü için "**int isim() {}**" gibi)
- Artık işlevimi ***.so** kitaplığına **derlemeye** hazırım:

```
> root mimariKarsilastirici2.C++
root [0] Processing mimariKarsilastirici2.C++...
Info in <TUnixSystem::ACLiC>: creating shared library
  /home/oc/Documents/HEP_Okulu/workDir/root/./mimariKarsilastirici2_C.so
```

- Kitaplığımı daha sonra tekrar **kullanmak** istediğimde:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir/root$ root -l
root [0] .L mimariKarsilastirici2_C.so
root [1] mimariKarsilastirici2()
```

# Kullanım

## Betiği *.so kitaplığına derlemek

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici.C | mimariKarsilastirici2.C

```
#include <TFile.h>
#include <TNtuple.h>
#include <TH2.h>
#include <TProfile.h>
#include <TCanvas.h>
#include <TFrame.h>
#include <TROOT.h>
#include <TSystem.h>
#include <TRandom.h>
#include <TBenchmark.h>
#include <TCint.h>
#include <TGraph.h>
#include <TF1.h>

int mimariKarsilastirici2() {
   gROOT->Reset();

   // edit these parameters for other simulations -----------
   int noofbits=10;
   int noofiteration4rms=100, nob=100; // nob : no of bins in distro
   double sigma=0.0, centroid=1.0;
   const Int_t kUPDATE = 50;
   bool update=kTRUE;
   //-----------------------------------------------------

   int nop=pow(2.0, noofbits);
   int boyut = pow(2, 1*noofbits);
   double Iu[boyut]; // 2^noofbits current source
   int lastincremented=0;
   TCanvas *c1 = new TCanvas("c1","Binary vs Thermometer",1200,850);
   TPad *pad1 = new TPad("pad1","Unit Current Source",0.0,0.0,1.0,1.0);
   pad1->Draw();
   pad1->Divide(5, 4, 0.001, 0.001);
   pad1->cd(5)->SetFillColor(47);  pad1->cd(5)->SetGrid() ;
   pad1->cd(6)->SetFillColor(47);  pad1->cd(6)->SetGrid() ;
   pad1->cd(7)->SetFillColor(47);  pad1->cd(7)->SetGrid() ;
   pad1->cd(8)->SetFillColor(47);  pad1->cd(8)->SetGrid() ;
   pad1->cd(9)->SetFillColor(47);  pad1->cd(9)->SetGrid() ;
   pad1->cd(10)->SetFillColor(47); pad1->cd(10)->SetGrid();
   pad1->cd(11)->SetFillColor(21); pad1->cd(11)->SetGrid();
   pad1->cd(12)->SetFillColor(21); pad1->cd(12)->SetGrid();
   pad1->cd(13)->SetFillColor(21); pad1->cd(13)->SetGrid();
   pad1->cd(14)->SetFillColor(21); pad1->cd(14)->SetGrid();
   pad1->cd(15)->SetFillColor(21); pad1->cd(15)->SetGrid();
   pad1->cd(16)->SetFillColor(21); pad1->cd(16)->SetGrid();

   pad1->cd(1);
```

Ln 1, Col 1 INS

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici.C | mimariKarsilastirici2.C

```
        if (i>0 && i<nop) {
            b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
            b_rms_dnl->SetBinContent(i-1, b_rms_dnl->GetBinContent(i-1)+(b_dnl[i-1]-Iu[0
        }
        b_dac_inl->SetBinContent(i, b_inl[i]-Iu[0]);
        if (i>0) b_dac_dnl->SetBinContent(i-1, b_dnl[i-1]);
    }
    pad1->cd(12);
    b_dac_inl->Draw();
    pad1->cd(13);
    b_dac_dnl->Draw();
    if (update) c1->Update();
    pad1->cd(17);
    delta_dac->Draw();
    pad1->cd(18);
    b_cumulatif->Draw();
    pad1->cd(11);
    b_dac_ter->Draw();

   } // end loop

    for (int i=0 ; i<nop ; i++) { // calculate RMS
        rms_dac->SetBinContent(i, sqrt(rms_dac->GetBinContent(i)/noofiteration4rms));
        if (i<nop-1) rms_dnl->SetBinContent(i, sqrt(rms_dnl->GetBinContent(i)/noofiterat
        rms_inl->SetBinContent(i, sqrt(rms_inl->GetBinContent(i)/noofiteration4rms));
        b_rms_dac->SetBinContent(i, sqrt(b_rms_dac->GetBinContent(i)/noofiteration4rms))
        if (i<nop-1) b_rms_dnl->SetBinContent(i, 1.55-3*(-Iu[0]/2+sqrt(b_rms_dnl->GetBin
        b_rms_inl->SetBinContent(i, sqrt(b_rms_inl->GetBinContent(i)/noofiteration4rms))
    }
    pad1->cd(8);
    rms_dac->Draw();
    pad1->cd(9);
    rms_inl->Draw();
    pad1->cd(10);
    rms_dnl->Draw();
    pad1->cd(14);
    b_rms_dac->Draw();
    pad1->cd(15);
    b_rms_inl->Draw();
    pad1->cd(16);
    b_rms_dnl->Draw();
    if (update) c1->Update();

   gBenchmark->Stop("binary_vs_thermometer");
   return 0;
}
```

Ln 1, Col 1 INS

- Betiğin **isim taşıdığına** ve kullanılan kitaplığın başlık kütüklerinin (header) **içerildiğine** ve işletim sistemine bir **tamsayı döndürüldüğüne** dikkat edin

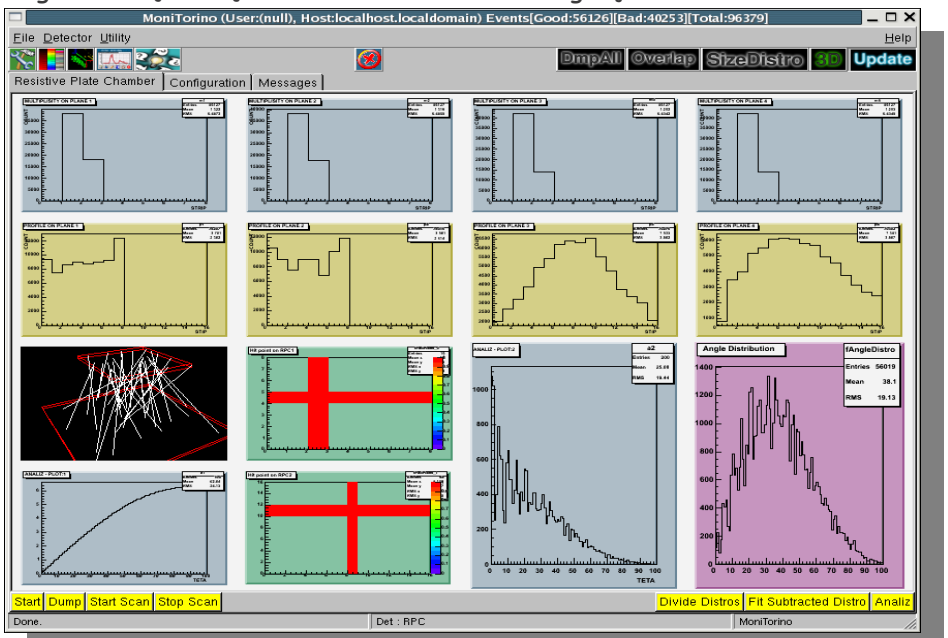
# Kullanım

## Bağımsız derlemek (uygulama yazmak)

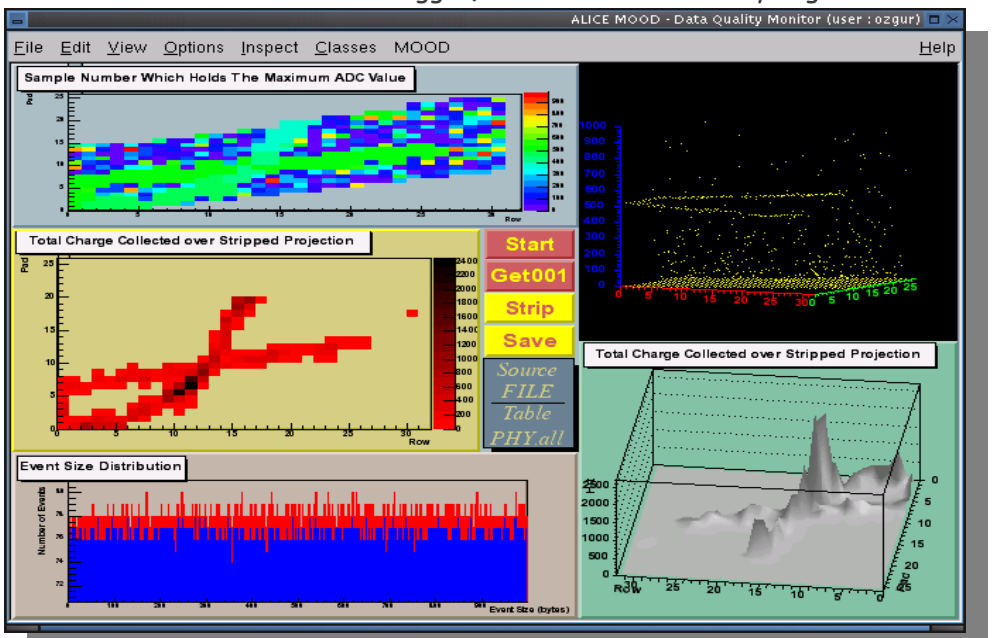
- Mimari karşılaştırıcım ***.so*** haline geldiğinde, ROOT ortamında "**.L**" ile yüklenebilecek ve sanki bir **ROOT komutuymuş gibi** çalıştırılabilecektir
- Daha **yüksek hızla** çalısacaktır
- ROOT' un **varlığına** ihtiyaç duyacak ve **ancak** uygun bir ROOT kitaplığı ile oluşturulabilecek ve çalıştırılabilecektir.

- **Halbuki** ROOT ortamına ihtiyaç duymaksızın, ROOT' u bir **kitaplık olarak** kullanmak (tıpkı *cizgi* ve *ok* kitaplığını kullandığımız gibi) ve **kendi ayakları üzerinde duran** (standalone) bir **uygulama** (application) geliştirmek de mümkün.

Canlı veri izleyici ve analiz yazılımı (MoniTorino) eğitim amacı ile, kozmik parçacıkların dünyanın yüzey normali ile yaptıkları açı dağılımını ölçmek için Torino Üniversitesi'nde geliştirildi.




CERN'deki ALICE deneyinde halen kullanılmakta olan DQM yazılımının (MOOD – Monitor of On-line Data and Detector Debugger) ilk sürümü bir TPC olayını gösterirken.



# Kullanım

## Bağımsız derlemek (ilk örneğin uygulama hali)

ilkUygulama.cxx | TTPLOTist.cxx | TTPLOTist.h

```
1 #include "TApplication.h"
2 #include "TCanvas.h"
3 #include "TLine.h"
4 #include "TF3.h"
5
6 int main(int argc, char **argv)
7 {
8     // ROOT uygulama nesnesi yaratiliyor
9    TApplication benimGuzelUygulamam("App", &argc, argv);
10
11    // Bir pencere (veya kanvas) olusturup, isletim sisteminin pencere
12    // yönecisi arasinda signal-slot haberlesmesini baslat
13   TCanvas *kanvas = new TCanvas("kanvas", "Benim Guzel Kanvasim", 500, 400);
14   kanvas->Connect("Closed()", "TApplication", &benimGuzelUygulamam, "Terminate()");
15
16    // Komut örnegimizde yaptigimizin aynisini yapiyoruz.
17   TF3 *islev = new TF3("islev","sin(cos(x)+sin(y)+cos(z))",-10,10,-10,10,-10,10);
18
19    // Nesnenin, kendisini görünür kilan üye islevini çagir
20   islev->Draw();
21
22    // Kanvas nesnesinin, üzerindeki nesnelerin görünürlük
23    // durumunu yenileyen üye islevini çagir
24   kanvas->Update();
25
26    // Uygulamayi kendi halinde çalismaya birak (GUI event loop' a sok)
27   benimGuzelUygulamam.Run();
28
29   // Sonlandirildiginda isletin sistemine basari isareti göndererek çik
30   return 0;
31 }
```

Başlıkları ekle

Alışılagelmiş C/C++ başlangıcı

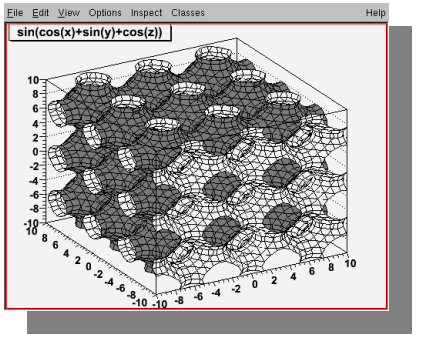
Ln 13, Col 69 INS

# Kullanım

## Bağımsız derlemek (ilk örneğin uygulama hali)

- Uygulama geliştirmede kullanılacak en seçkin derleme yöntemi **Makefile** kullanmaktır; fakat biz burada -bu sayfadaki bilgi Makefile kullanılması durumunda da hala geçerli olmakla birlikte- sadeliği korumak için kullanmayacağız.
- Bir önceki sayfadaki programı derlemek için:
  - **g++ -L/usr/lib/root -lCore -lCint -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -lMatrix -lPhysics -lz -pthread -lm -ldl -rdynamic -pthread -m64 -I/usr/include/root -L/usr/lib/root -lCore -lCint -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -lMatrix -lPhysics -lz -lGui -pthread -lm -ldl -rdynamic ilkUygulama.cxx -o ilkUygulama**
- Yukarıdaki uzun komutu hatırlamak zor olduğu için:
  - **root-config**: ROOT kullanıcılarının hayatını kolaylaştıran bir komut satırı uygulaması (çoğu kitaplık böylesi yardımcı programlara sahiptir)
  - Çağırana, ROOT kitaplığını kullanan programlarınızı **derlemek** için **gerekli içeriği** döndürür
  - Çoğunlukla **kaçış simgesi** (escape symbol) olarak bilinen tek tırnaklar " ` " arasında kullanılır
  - **g++ `root-config --glibs --cflags` ilkUygulama.cxx -o ilkUygulama**

```
oc@olmak2:~/HPFBU/root$ ./ilkUygulama &
[1] 16811
oc@olmak2:~/HPFBU/root$ _
```



# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak

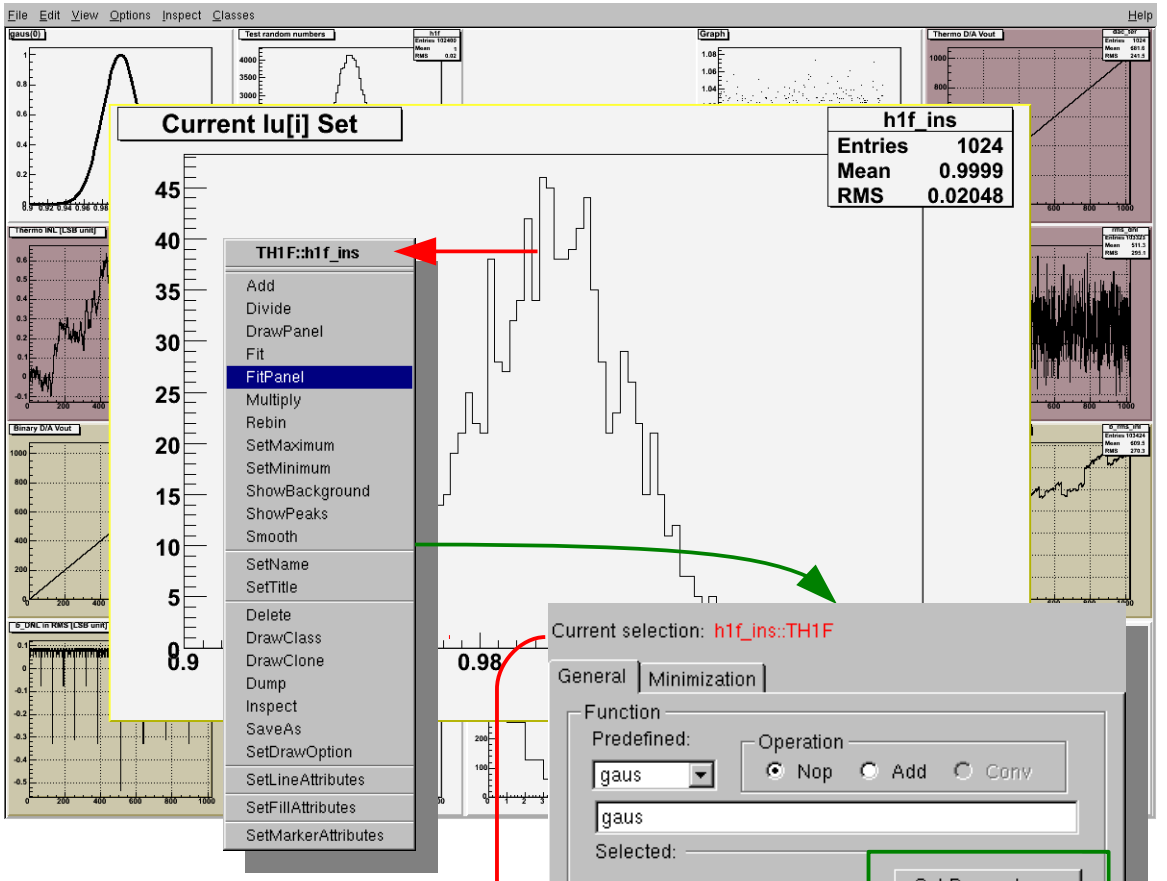

- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*

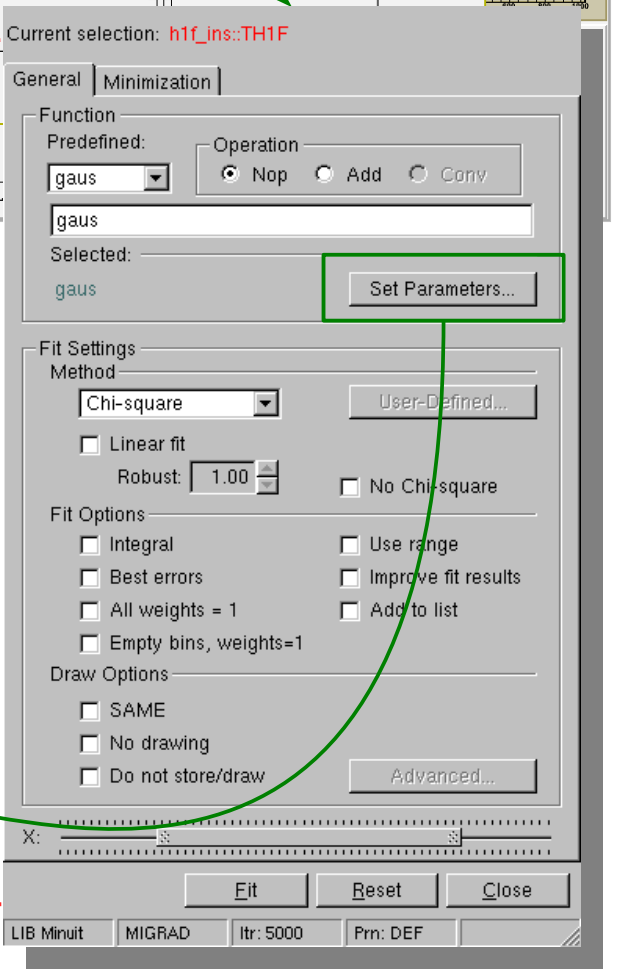
# Kullanım

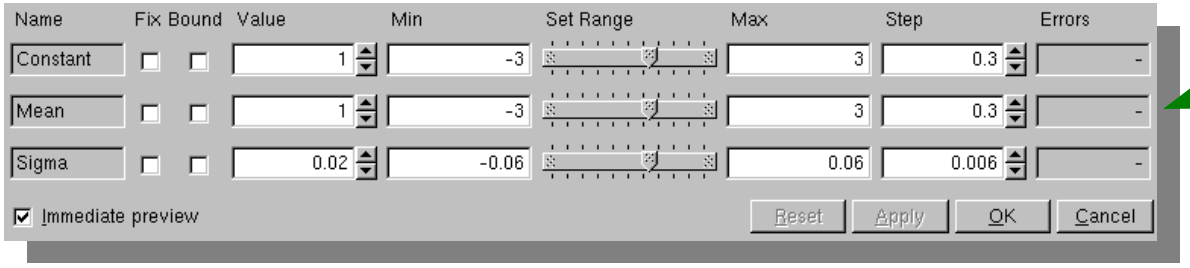
## Kullanıcı ile etkileşim

- Açılan tüm pencereler ve bu pencerelerde görünen **HERŞEY ama HERŞEY** ya ROOT class kitaplığından yaratılmış **NESNE**LERDİR (histogram, eksen, başlık, alt-pencere v.b.) ya da ROOT class kitaplığı aracılığı ile ulaşılabilecek başka kitaplıklardan yaratılmış başka nesnelerdir (x11 kitaplığından yaratılan nesneler ve pencere yönetici işlevleri gibi).
- **Sağ tıklama**, öbekte (heap) oluşturulmuş bu nesnelerin **üye işlevler**inden (class member function) bazılarına **erişim sağlar**.



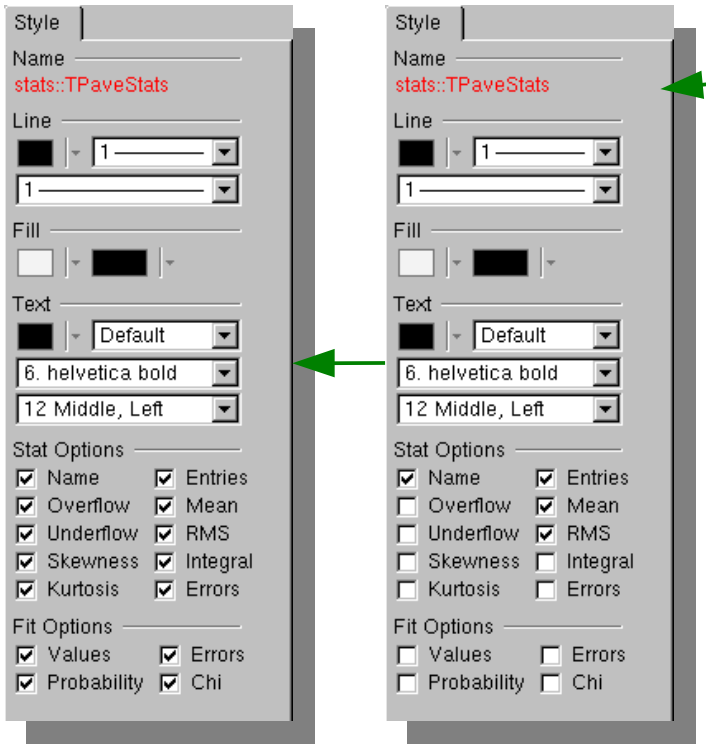

- ***Ön tanımlı** uyum işlevleri (fit function)*
- *Pekçok değişken*
- *Hesap aralığı*



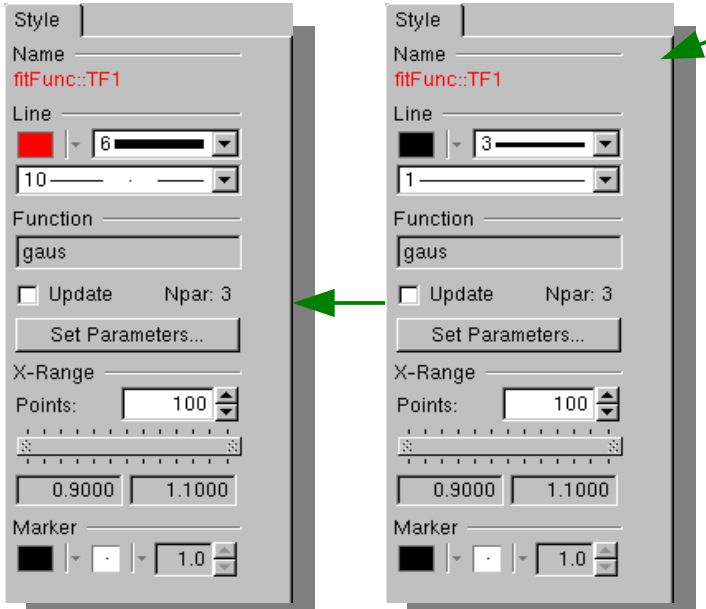
# Kullanım

## Kullanıcı ile etkileşim



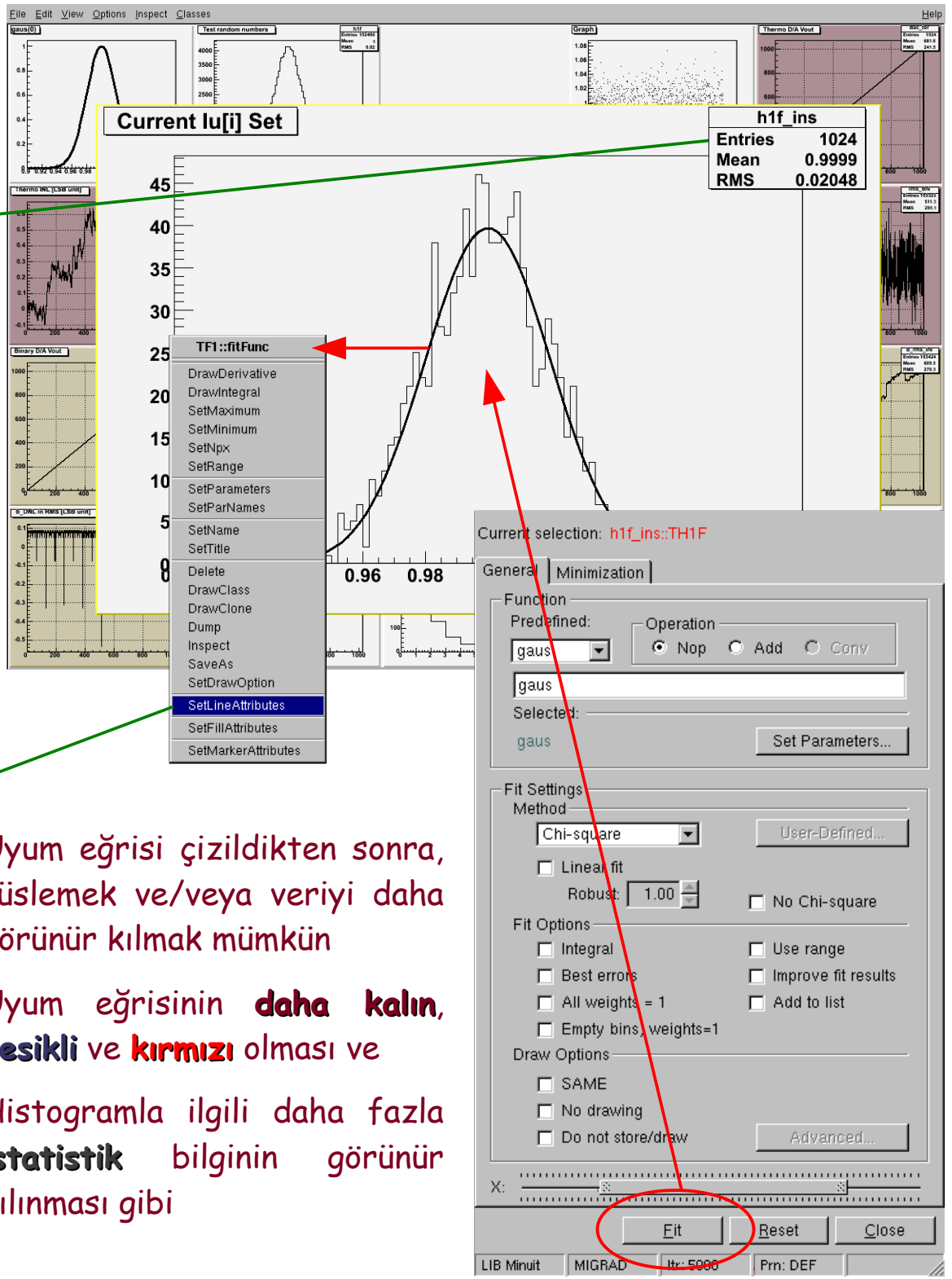










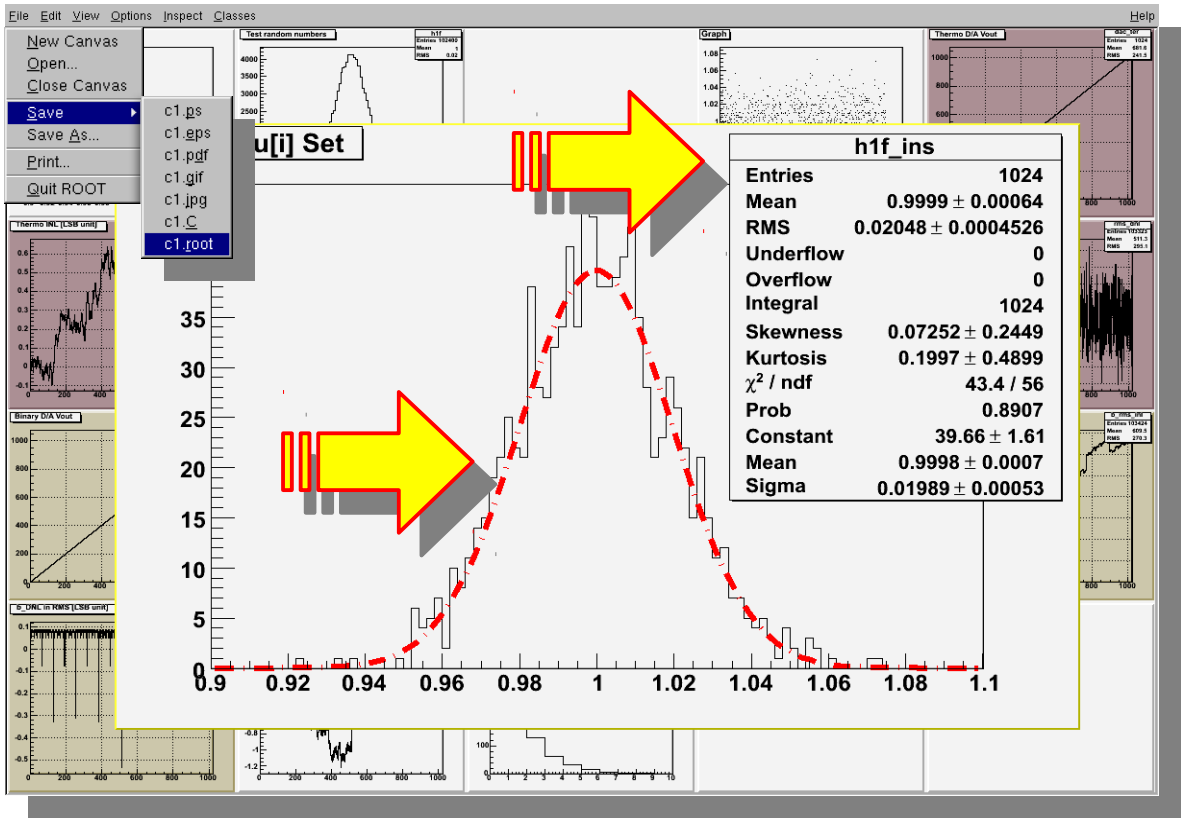



- Uyum eğrisi çizildikten sonra, süslemek ve/veya veriyi daha görünür kılmak mümkün
- Uyum eğrisinin **daha kalın**, **kesikli** ve **kırmızı** olması ve
- Histogramla ilgili daha fazla **istatistik** bilginin görünür kılınması gibi

# Kullanım

## Kullanıcı ile etkileşim

- Veri çözümleyen (analiz) yazılımımız (betik) başarı ile çalıştı, sonuç üretti, kullanıcı sonuç grafikleri üzerinde gerekli düzeltmeleri yaptı veriyi **daha okunaklı** hale getirdi.
- Yapılan **çalış**mayı **sakla**maya geldi sıra = **kaydet**mek !
- Farklı yollar mevcut; eldeki nesne:
  - Bir ***.C kaynak kütüğü** olarak saklanabilir
    - Bu kütüğü açmak, root ile **tekrar yorumlamayı** gerektirir (ör. "**root -l k.C**")
  - Bir ***.root kütüğü** olarak saklanabilir
    - Bu kütüğün içeriğine ulaşmak, bir **TBrowser nesnesi** ile doğrudan mümkündür (ör. "**TBrowser a**")
  - **Resim olarak** da saklanabilir:
    - ps, eps, gif v.b.



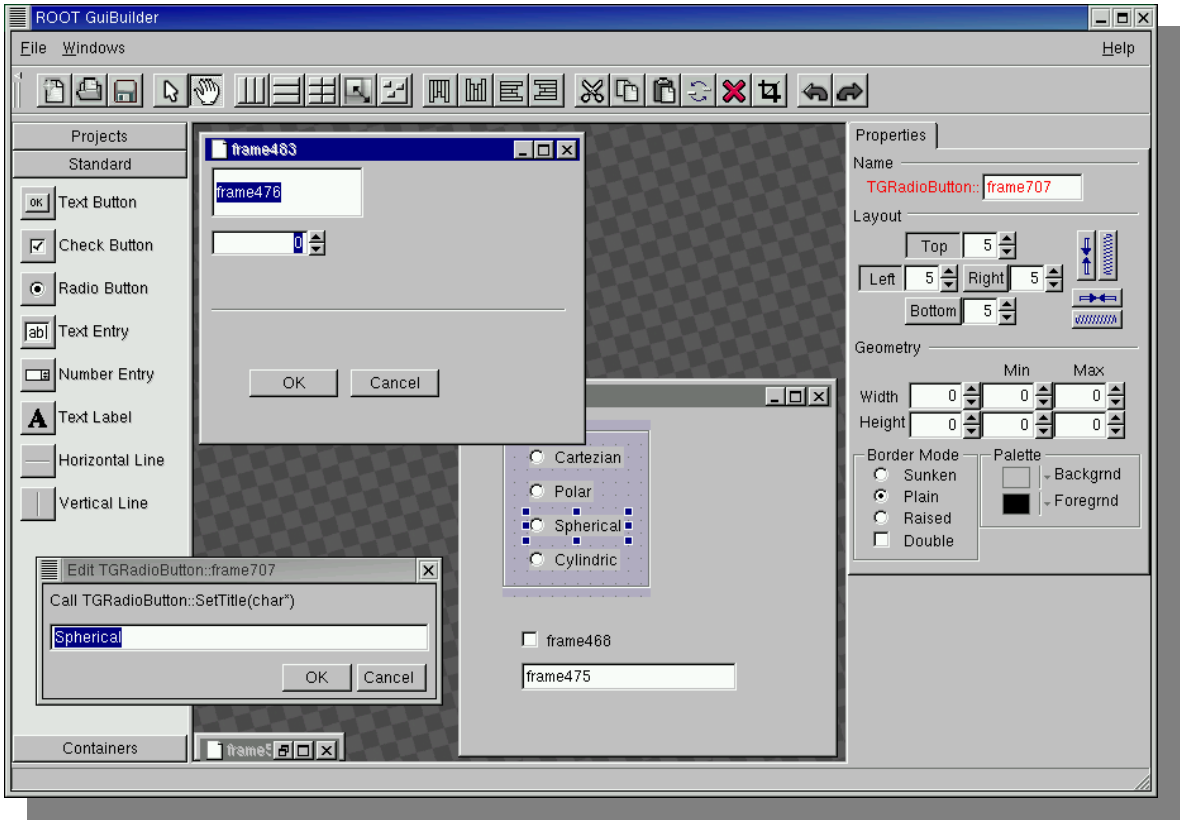

- Uyum eğrisi **daha kalın**, **kesikli** ve **kırmızı**
- Histogramla ilgili daha fazla **istatistik** bilgi görülüyor

# Örnek

## Kullanıcı arayüzü yaratmak

- Kullanıcılar ROOT GUI class kitaplığını kullanarak, yazdıkları uygulamalara grafik kullanıcı arayüzü (GUI veya HI) yazabilirler.
- Bunu yapmanın temelde iki yolu vardır ve çoğunlukla dönüşümlü olarak beraber kullanılırlar:
  - **Kaynak** yazarak ve ya
  - **TRootGuiBuilder** class' ının bir nesnesini kullanarak:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir/root$ root -l
root [0] TRootGuiBuilder a
root [1] _
```




- **Görsel** ve **etkileşimli** olarak hazırlanan kullanıcı arayüzü, ROOT betiği olarak, ***.C** uzantısı ile kaydedilir ve asıl işi yapacak kaynağın geliştirilmesine bu kütük üzerinde devam edilir.
- **Eğitici: yap, kaydet, oku !!**
- Zaman kazandırıcı: ROOT GUI class' larını **ezbere bilme gerekliliği yok**
- **Sık sık kaydet, arada göçebilir !**

# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak


- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*

# ROOT'a Rağmen ROOT'la Çalışmak

## Kullanım kılavuzu

- Yeni bir kitaplık ile tanışmak dört ayak üzerine oturur : (sanırım)
  i. Kullanım kılavuzu (user manual)
  ii. Kaynak belgelendirmesi (source code documentation)
  iii. Kitaplığın kullanımına örnekler (tutorial, test, example, v.b.)
  iv. Sizin o örnekleri çalığıp kendi uygulamalarınızı yazmaya başlamanız

- **Şanslıyız: ROOT bunların hepsine sahip !! ROOT' la tanışmak zor değil !!**

- **Kullanım kılavuzu**, kitaplığın **hangi düşüncelerle ve nasıl** oluşturulduğunu, kurulum ve "**merhaba dünya**" başlangıcının nasıl yapılacağını gösterir.
  - Hızlı değişmez, **durağandır**.
  - En başta **sadece bir kez** düzgün ve sindirerek çalışılmalı.
  - Pek çok geliştirici için her yeni sürümde, **yüzeysel göz gezdirmek yeterli**

- Bu derste anlatılanlardan çok **daha ayrıntılı bilgiye**, **akıcı bir dil** ile yazılmış kullanım klavuzundan ulaşmak mümkün

# ROOT'a Rağmen ROOT'la Çalışmak

## Kullanım kılavuzu

- Yeni bir kitaplık ile tanışmak dört ayak üzerine oturur : (sanırım)
  i. Kullanım kılavuzu (user manual)
  ii. Kaynak belgelendirmesi (source code documentation)
  iii. Kitaplığın kullanımına örnekler (tutorial, test, example, v.b.)
  iv. Sizin o örnekleri çalığıp kendi uygulamalarınızı yazmaya başlamanız

- Şans… T' l… zor değil !!

- Kulla… luğunu, kurul… l yap… erir.
  - Hızlı değişmez, **durağandır**.
  - En başta **sadece bir kez** düzgün ve sindirerek çalışılmalı.
  - Pek çok geliştirici için her yeni sürümde, **yüzeysel göz gezdirmek yeterli**

Pekiyi, bu zaman kaybı neden ?

- Bu derste anlatılanlardan çok **daha ayrıntılı bilgiye**, **akıcı bir dil** ile yazılmış kullanım klavuzundan ulaşmak mümkün



* Lolcat' ten

# ROOT'a Rağmen ROOT'la Çalışmak

## Kullanım kılavuzu

ROOT kullanım kılavuzu ~500 sayfadır ! Üstelik ...

... sadece bir "dil" öğretir, o dilde neyin/nasıl söyleneceğini değil !



* Lolcat' ten

# ROOT'a Rağmen ROOT'la Çalışmak

## $ROOTSYS/tutorials dizininin etkin işlevi

- **Betik** yazarken **ölümle kalım** arasındaki çizgi
- Öğrenilecek/çalışılacak bir şey **değil**, ROOT' arken devamlı olarak başvurulacak en düşük seviyeli (asıl olana en yakın mesafede) öğretmen

- **Yeni bir kitaplık kullanmak**, özellikle C/C++ gibi kitaplık desteği olmadan herşeyin zor olduğu dillerde, **yeni bir dil öğrenmek**tir.
- Dil **konuş**ularak ve/veya **kullan**ılarak öğrenilir.
- Bu dizin, birlikte ROOT' ça **konuşmaya başlanacak arkadaşlar**ımızın yaşadığı yer

- ROOT ile yeni tanışmada bu dizindeki **TÜM ama TÜM alıştırmaları**, açın, ne yapmak istediğini yüzeysel olarak anlayın ve çalıştırın.
- Çünkü:
  - Alıştırmaların ne kadarının çalıştığını ne kadarının çalışmadığını kendi tecrübenize dayanarak öğrenmelisiniz – kullanacağınız **kitaplık mükammel mi** ?
  - Bu alıştırmalar, ileride yazacağınız uygulamalar için **başlangıç örnekleri** içeriyor.
  - Hangi *.C kütüğünün hangi işlevi yerine getirdiğini hatırlamanız gere**kmiy**or.
  - Sadece **"tutorial' da bu işi yapan bir alıştırma vardı, oradan başlayabilirim..."** diyebilmeniz gerekli

# ROOT'a Rağmen ROOT'la Çalışmak

## $ROOTSYS/test dizininin etkin işlevi

- **Uygulama** yazarken **ölümle kalım** arasındaki çizgi
- Öğrenilecek/çalışılacak bir şey **değil**, ROOT' arken devamlı olarak başvurulacak en düşük seviyeli (asıl olana en yakın mesafede) öğretmen

- **Yeni bir kitaplık kullanmak**, özellikle C/C++ gibi kitaplık desteği olmadan herşeyin zor olduğu dillerde, **yeni bir dil öğrenmek**tir.
- Dil **konuş**ularak ve/veya **kullan**ılarak öğrenilir.
- Bu dizin, birlikte ROOT' ça **konuşmaya başlanacak arkadaşları**mızın yaşadığı yer

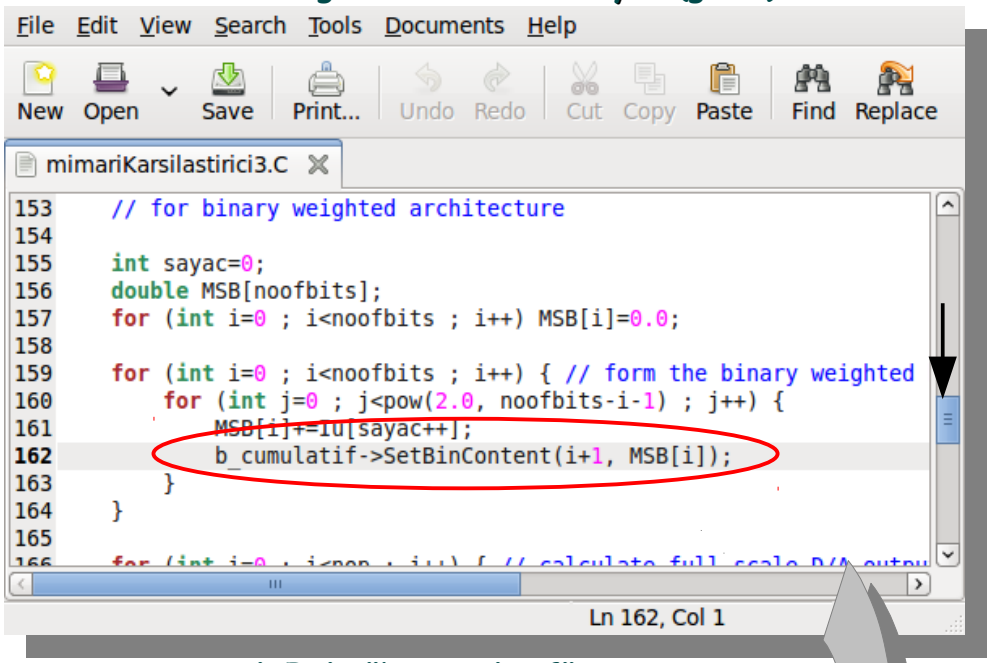
- ROOT ile yeni tanışmada bu dizindeki **TÜM ama TÜM uygulamaları**, açın, ne yapmak istediğini yüzeysel olarak anlayın, derleyin ve çalıştırın.
- Çünkü:
  - Uygulamaların ne kadarının çalıştığını ne kadarının çalışmadığını kendi tecrübenize dayanarak öğrenmelisiniz – kullanacağınız **kitaplık mükammel mi**?
  - Bu uygulamalar, ileride yazacağınız uygulamalar için **başlangıç örnekleri** oluşturuyor.
  - Hangi *.cxx (ve karşılık gelen *.h) kütüğünün hangi işlevi yerine getirdiğini hatırlamanız gerek**mi**yor.
  - Sadece **"test' te bu işi yapan bir alıştırma vardı, oradan başlayabilirim."** diyebilmeniz gerekli

# ROOT'a Rağmen ROOT'la Çalışmak

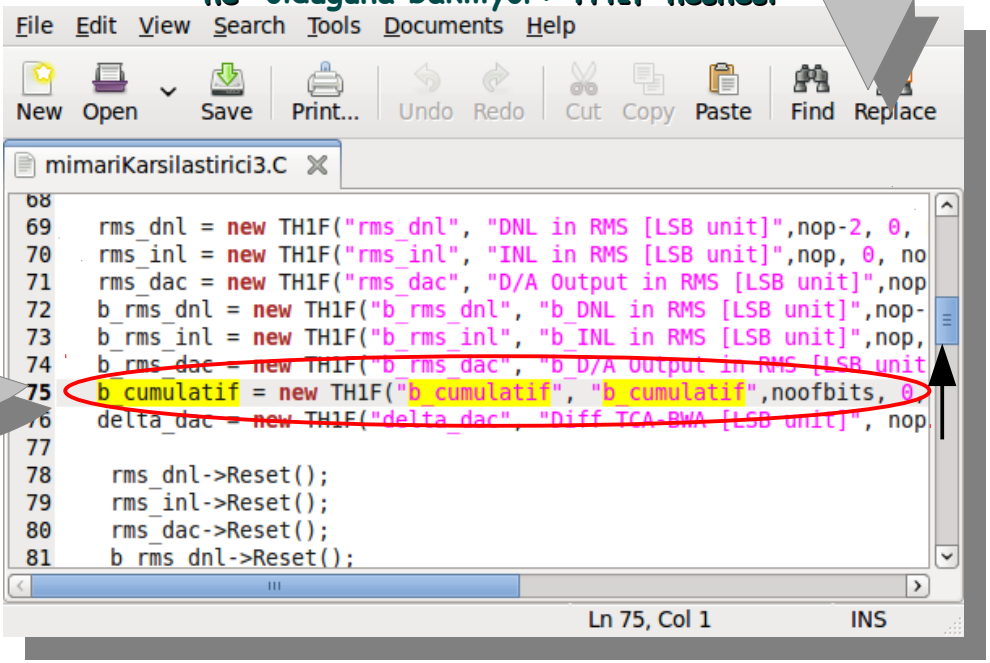
## HTML belgelendirmesi

- **Ölümle kalım** arasındaki çizgi
- Öğrenilecek/çalışılacak bir şey **değil**, ROOT' arken **devamlı olarak başvurulacak** en düşük seviyeli (asıl olana en yakın) belgelendirme
- **Canlı:** kullanılıyor olan **ROOT sürümünden doğrudan üretiliyor**, dolayısıyla:
  - **Kullanım klavuzları** gibi **durağan** (statik), dolayısıyla **eskimiş bilgilerle dolu** ve bu anlamda **kullanışsız** değil
  - Kullanılan kitaplığın **kaynağını okumak**la eşdeğer, **hataya en az açık** yöntem

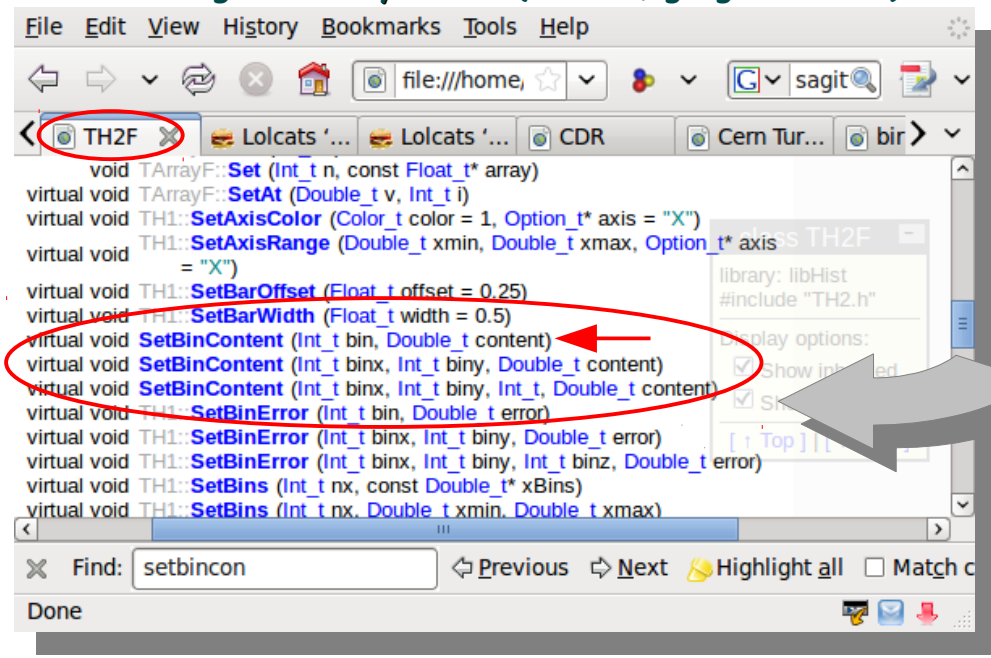
En sevdiğiniz **metin düzenleyici** (gedit)




**Ctrl-F** ile "b_cumulatif" aranıyor ve "ne" olduğuna bakılıyor: **TH1F nesnesi**




En sevdiğiniz **html yorumcu** (firefox, google-chrome)



# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak

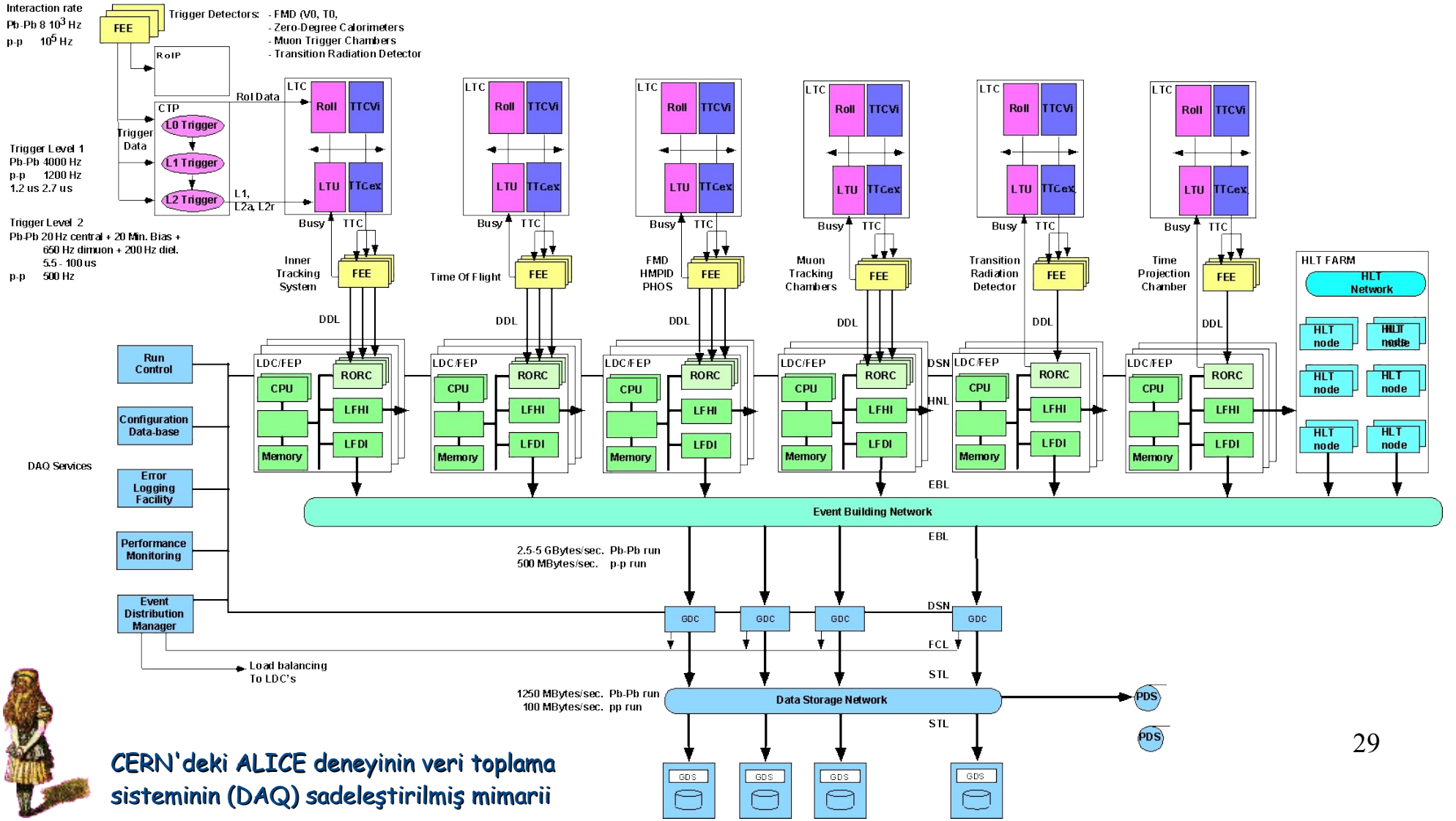

- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*

# Başka Kitaplıkların ROOT ile Kullanımı

## Örnek: CERN'deki ALICE deneyi için canlı veri izleyici (DQM) yazılımı

- **Büyük ölçekli** deneysel sistemler alt-sistemlerin **derin bir hiyerarşisi** şeklindedir
- Böylesi sistemler **farklı seviyelerde sürekli izlenmelidir** (ör: LDC ve GDC gibi). Bunun nedeni veri kalitesinin pekçok farklı değişkene bağlı olmasıdır (insan, hızlandırıcı, algıç denetim sistemleri gibi)
- Veriye erişim, **Veri Toplama Takımı** tarafından geliştirilen bir **C/C++ kitaplığı** aracılığı ile sağlanır

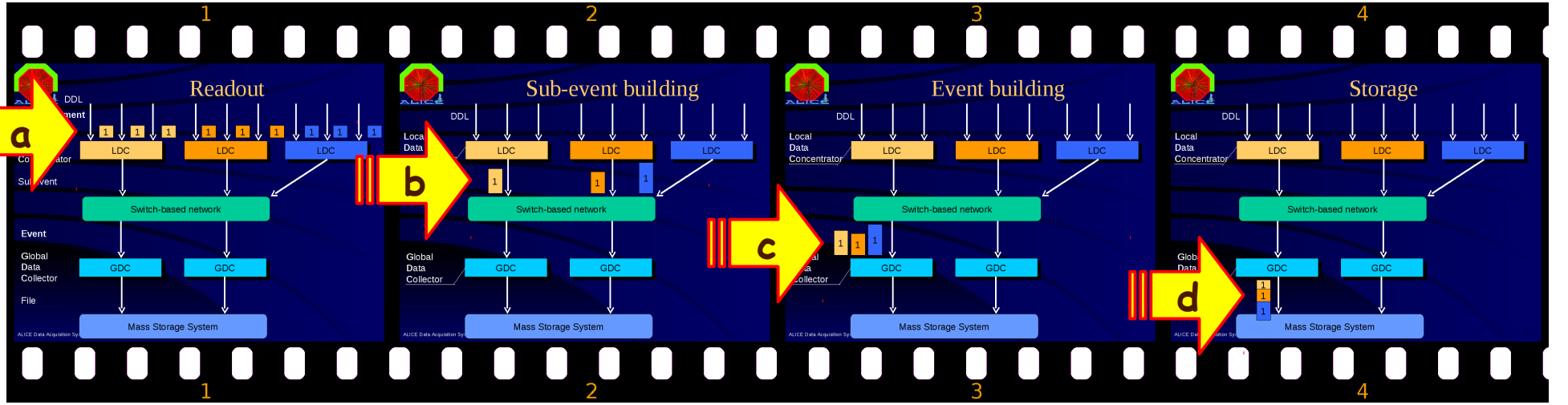





CERN'deki ALICE deneyinin veri toplama sisteminin (DAQ) sadeleştirilmiş mimarii

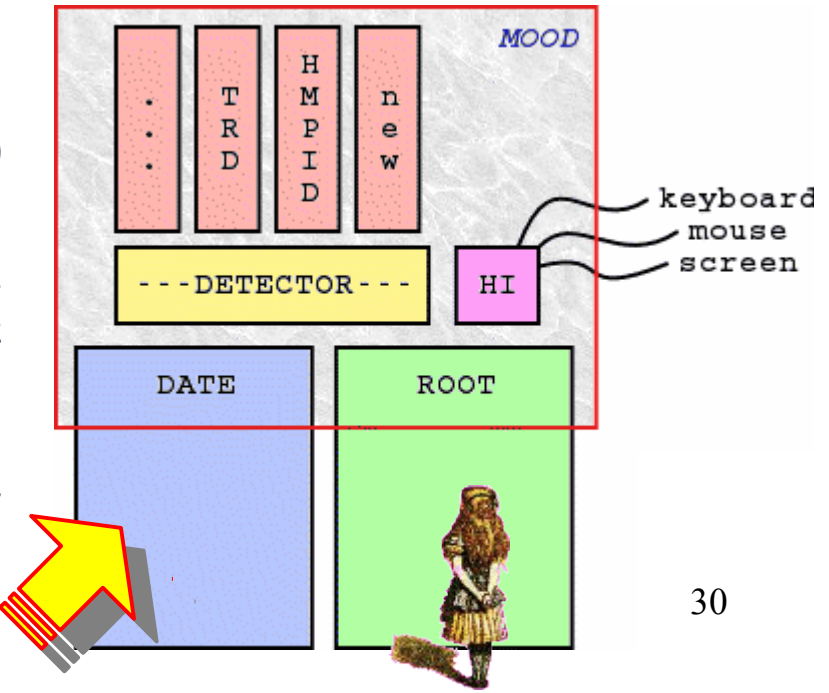
# Başka Kitaplıkların ROOT ile Kullanımı

## Veri toplama sisteminin sadeleştirilmiş işleyişi




* Pierre Vande Vyvre'den veri toplama sisteminin daha da sadeleştirilmiş hali)

- Algıçokuyanlar farklı alt-algıçları okuyor: **yükler** (a)
- Yükler, veri toplama sisteminde aşağıya LDC'lere (Local Data Collector) doğru gelerek birleşiyor: **olay bileşenleri** (b)
- Olay bileşenleri aşağıya GDC'lere (Global Data Concentrator) doğru ilerleyerek birleşiyor **alt-olaylar** (c)
- Alt-olaylar biraraya gelerek **super-olay** oluşturuyorlar ve **kalıcı veri saklama** birimlerine daha sonra ayrıştırılmak (analiz) üzere gönderiliyorlar (d)
- **Görevimiz**, veriye erişim için **DATE**, görselleştirme ve etileşimlileştirme gibi diğer herşey için **ROOT** kitapliklarını kullanan bir **canlı veri izleyici** yazılımı geliştirmek



# Başka Kitaplıkların ROOT ile Kullanımı

Ayrı bir kitaplığın sağladığı işlevlerle veriye erişim sağlamak

```
oc@olmak-x200:~$ root -l
root [0] gSystem->Load("${DATE_MONITOR_DIR}/{DATE_SYS}/libmonitor.so");
root [1] monitorSetDataSource(":"); /* Yerel canli izlemeyi aç*/
root [2] int *event;
root [3] monitorGetEventDynamic( &event );
root [4] printf( "%08x %08x \n", event[0], event[1] );
0000878c da1e5afe
Root [5] .q
oc@olmak-x200:~$ _
```

*DATE Olay Biçimi*

0 – Olay boyutu (*0000878c*)
1 - DATE olay imzası (*da1e5afe*)
2 – Başlık boyu
3 – Taban olay başlık sürümü
4 – Olay çeşidi
5 – Olayla ilgili deneme sayısı
6 – Eşsiz olay kimliği
7 – Olayın 2. seviye tetiği
8 – Olayın algıç kimliği
9 – Olay özellikleri
10 – LDC kimliği
11 – GDC kimliği
12 – Olay zamanı
*< n. alet başlığı >*
*< n. aletin verisi >*
*< (n-1). alet başlığı >*
*< (n-1). aletin verisi >*
...

Büyük olay başlığı (super event header)

Alt-algıç verisi (yük, payload)

Bir başka yük

*Özet:*

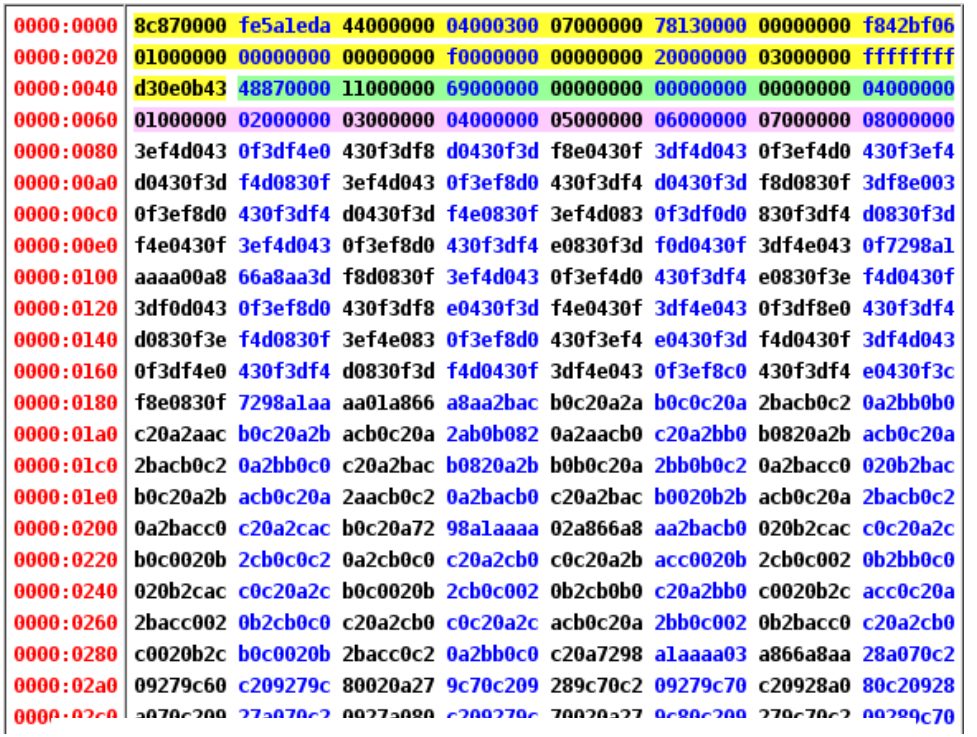
```
Root [0] → İzleme kitaplığını yükle
Root [1] → Veri kaynağını seç
Root [2] → Verinin göçertileceği değişken
Root [3] → Veriyi değişkene göçert
Root [4] → İlk iki “kelime”yi göster
Root [5] → Oturumu kapat
```

*Bu, kimliğinizde sizin hakkınızdaki bilgilerden bile daha fazlasıdır :)*

# Başka Kitaplıkların ROOT ile Kullanımı

## Algıç verisinin dil bilgisi kurallarına göre algılanması

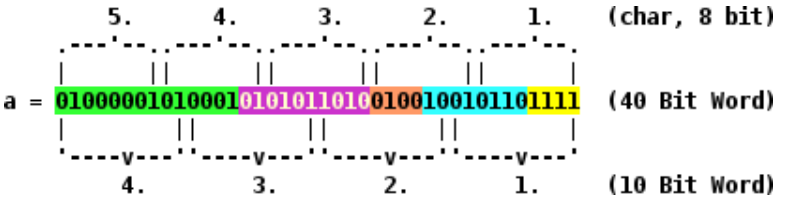
- Tamamı yaklaşık 60 milyon sekizli (Mbyte) olan ALICE deneyinin TPC (Time Projection Chamber) algıcının ürettiği verinin başlangıcı




- Sarı, yeşil ve pembe renklerle belirginleştirilmiş kısımlar algıcın ve veri toplama sisteminin farklı kısımlarınca üretilmiş veridir (olay / alet / hırdavat başlıkları)

- Geri kalan ise TPC tarafından üretilmiş, asıl ilgilendiğimiz **işlenmemiş veridir**

- Bir TPC "kelimesi" aşağıdaki gibi farklı biçimlerde bileşenlerine ayrıştırılabilir:
  - 10 bitlik 4 kelime
  - 8 bitlik 5 kelime
  - 40 bitlik tek bir kelime
  - Sırasıyla soldan sağa 4, 8, tekrar 4, 10 ve 14 bitlik 5 kelime (Biz bu seviyedeki bilgi ile ilgileniyoruz)

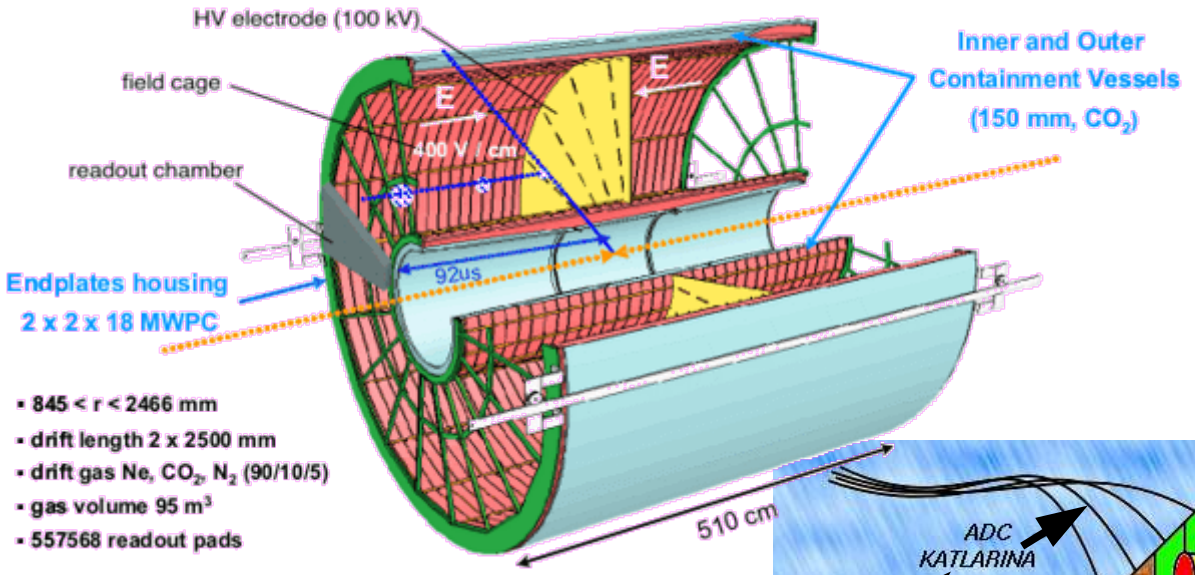



| Seq. | bin | dec | (10 Bit Words, trailer) |
|---|---|---|---|
| Channel. | 1111 | 15 | (yellow) |
| Hardware. | 10010110 | 150 | (light blue) |
| Pattern A | 0100 | 4 | (orange) |
| 10BitWord | 0101011010 | 346 | (magenta) |
| 2AAA Pat. | 01000001010001 | 4177 | (green) |

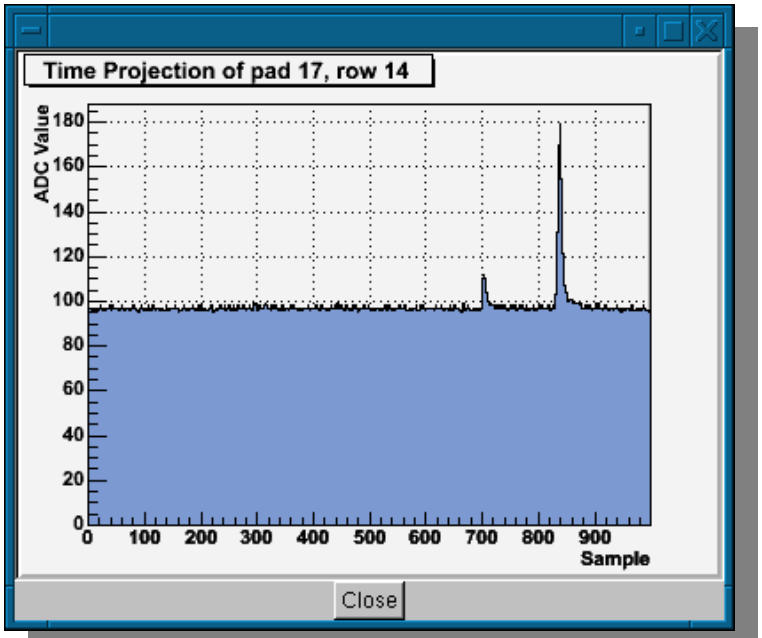
# Başka Kitaplıkların ROOT ile Kullanımı

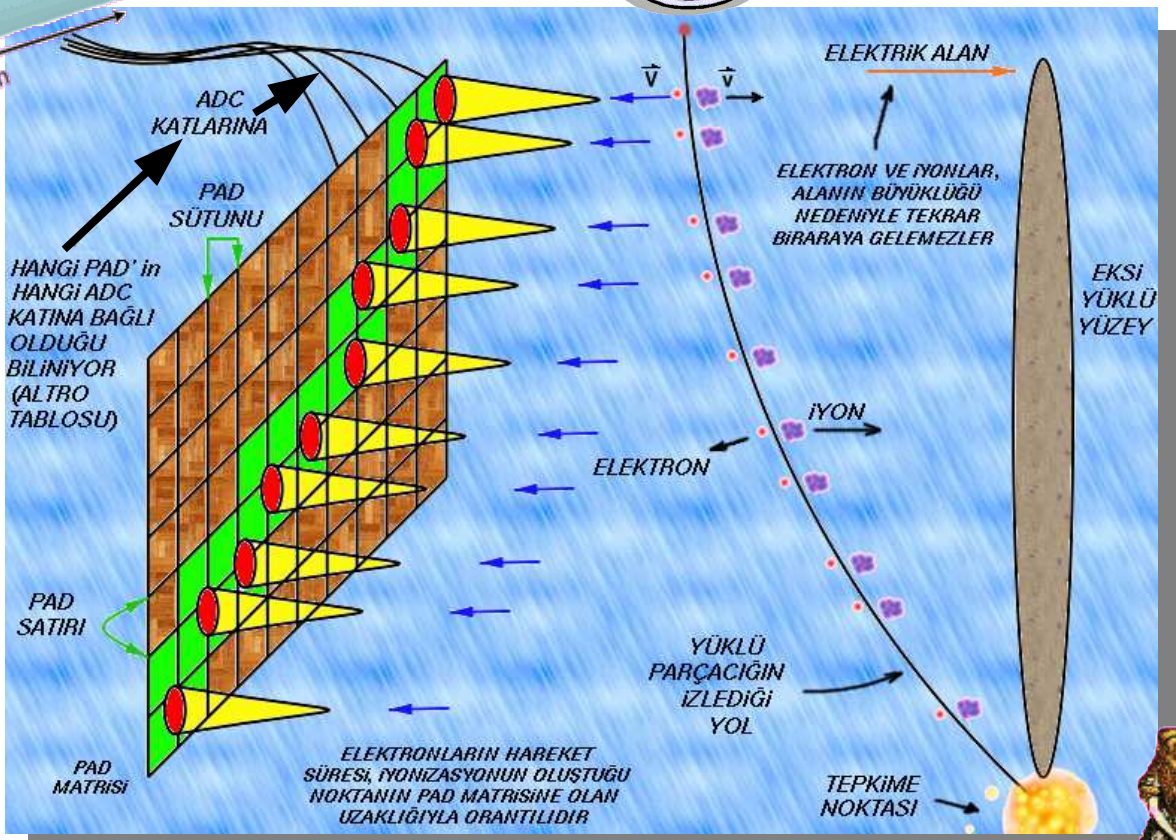
## Algıcın üretiği veriyi anlamak

- Alıştırmamız için, 3B bir izleyici (tracker) olan ALICE'in TPC algıcını seçelim




TPC algılama ilkesi

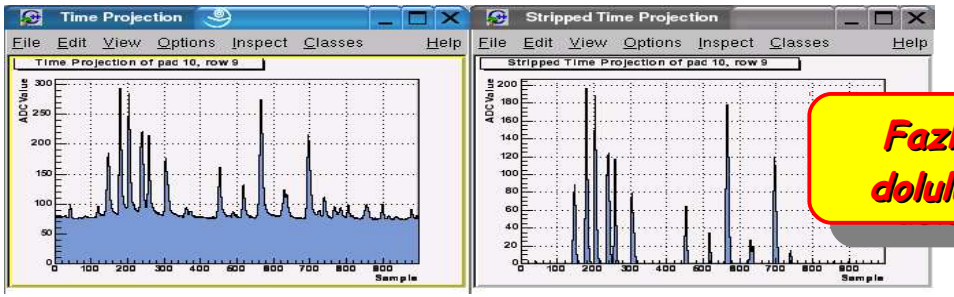


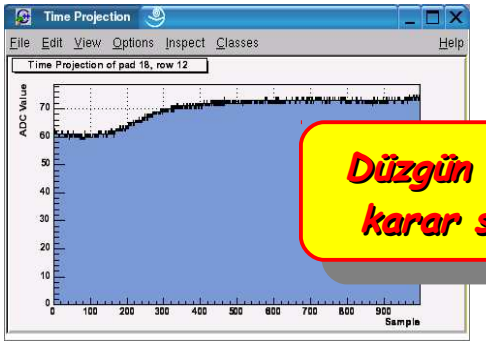
# Ne Görmeyi Bekliyoruz ?

## Bir canlı veri izleyiciden (DQM) ne istiyoruz ?

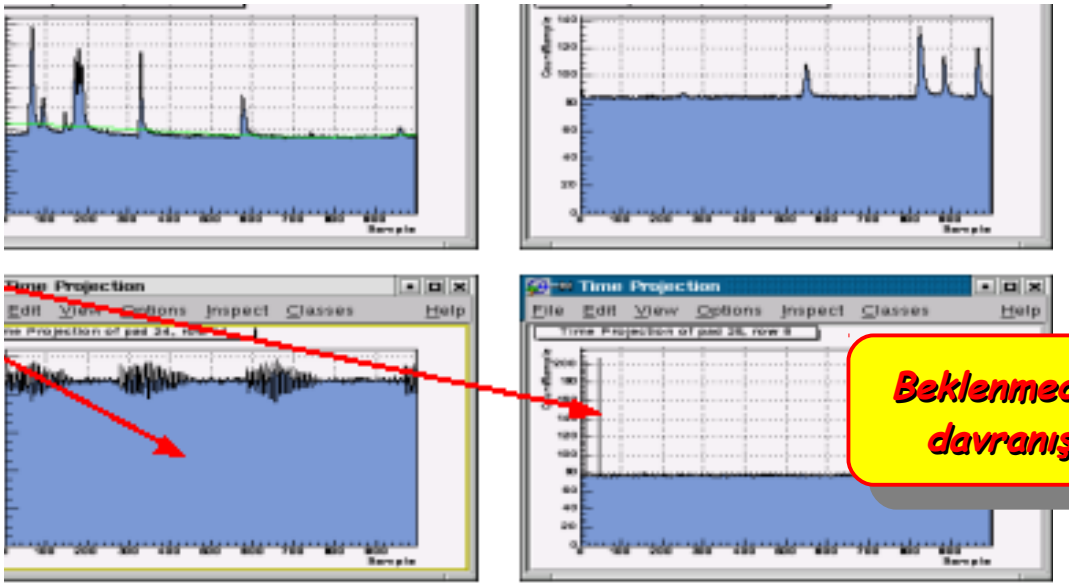
- **Elektronik** konular
  - **Takılı kalma** (Stack-at) hataları
  - **Ayarlanamaz** (non configurable) FE, v.b.
  - **Beklenmedik davranış**
- **Algıçla** ilgili konular
  - **Gaz** karışımı ve akışı
  - **Doluluk oranı** (occupancy), v.b.
  - **Beklenmedik davranış**
- **Genel**
  - Sıcaklık ve **güç kaynağındaki dalgalanmalar**
  - Tepelerin **biçimi** (peak formation)
  - **Karar** seviyesindeki (baseline) dalgalanmalar, v.b.
  - **Beklenmedik davranış**




Fazla doluluk




Düzgün olmayan karar seviyesi



- Hep aynı değeri üreten kanal
- Karar seviyesindeki aşina olunmayan bir dalgalanma

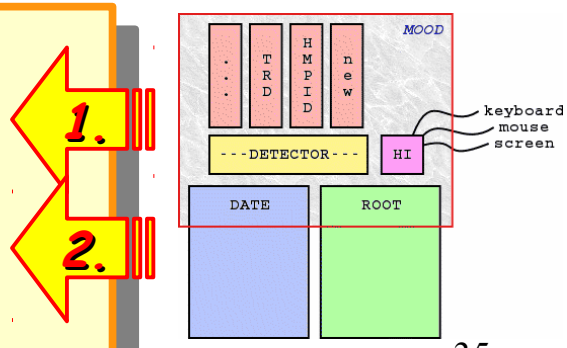
Beklenmedik davranış

# Uygulama Derleme

## Birden fazla kitaplık üzerine kurulu uygulama derleme

- Uygulama geliştirmede kullanılacak en seçkin derleme yöntemi **Makefile** kullanmaktır; fakat biz burada -bu sayfadaki bilgi Makefile kullanılması durumunda da hala geçerli olmakla birlikte- sadeliği korumak için kullanmayacağız.
- Bir önceki sayfadaki programı derlemek için:
  - `g++ -L/date_home -lbase -laz -lgui -thread -lm -lmst -ldynamic -lnonStand -lshift -lcustomDet -lmntrno {...} ikinciUyg.cxx -o ikinciUyg`
- Yukarıdaki uzun komutu hatırlamak zor olduğu için:
  - **date-config**: DATE kullanıcılarının hayatını kolaylaştıran bir komut satırı uygulaması (çoğu kitaplık böylesi yardımcı programlara sahipti; hatırlayın)
    - Çağırana, DATE kitaplığını kullanan programlarınızı **derlemek** için **gerekli içeriği** döndürür
    - Çoğunlukla **kaçış simgesi** (escape symbol) olarak bilinen tek tırnaklar " ` " arasında kullanılır
  - `g++ `date-config --glibs --cflags` ucuncuUyg.cxx -o ucuncuUyg`

```
g++
`root-config --glibs --cflags`
`date-config --glibs --cflags`
ucuncuUyg.cxx -o ucuncuUyg
```



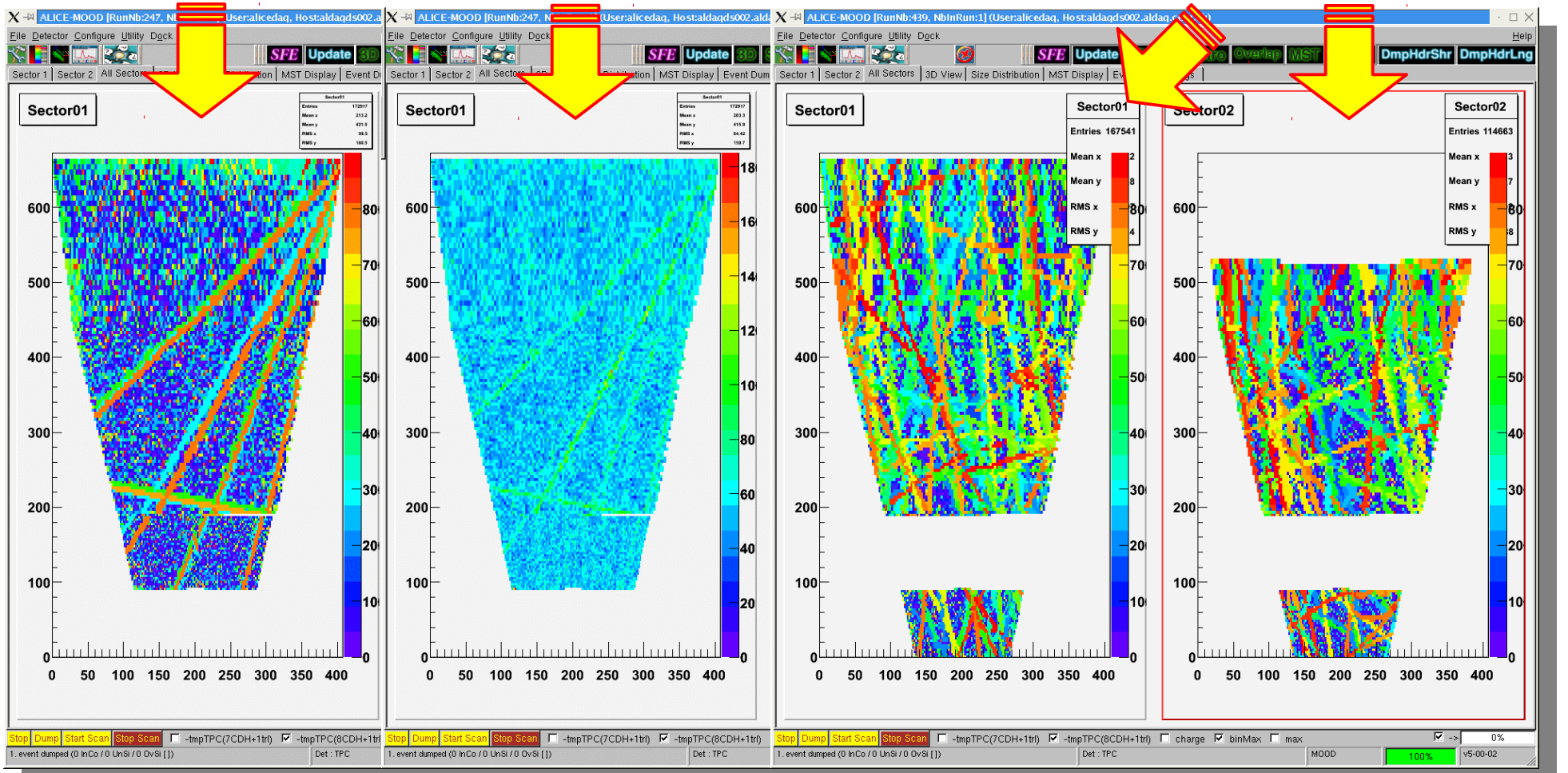
# Başka Kitaplıkların ROOT ile Kullanımı

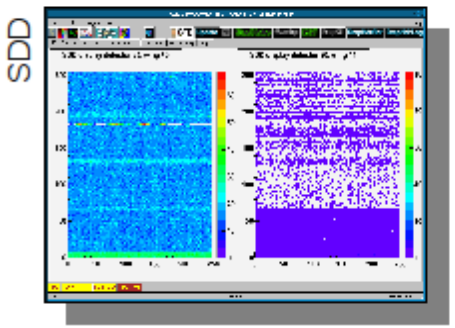
## Kullanıma hazır uygulamanın teslimi

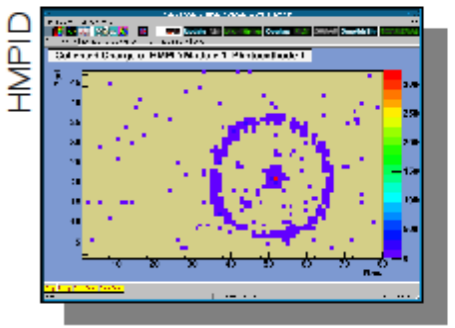
İki farklı lazer olayı

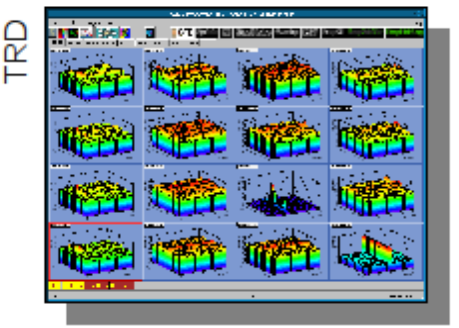
Aynı lazer olayı farklı algoritmalarla görünür kılınmış

Bir kozmik olay



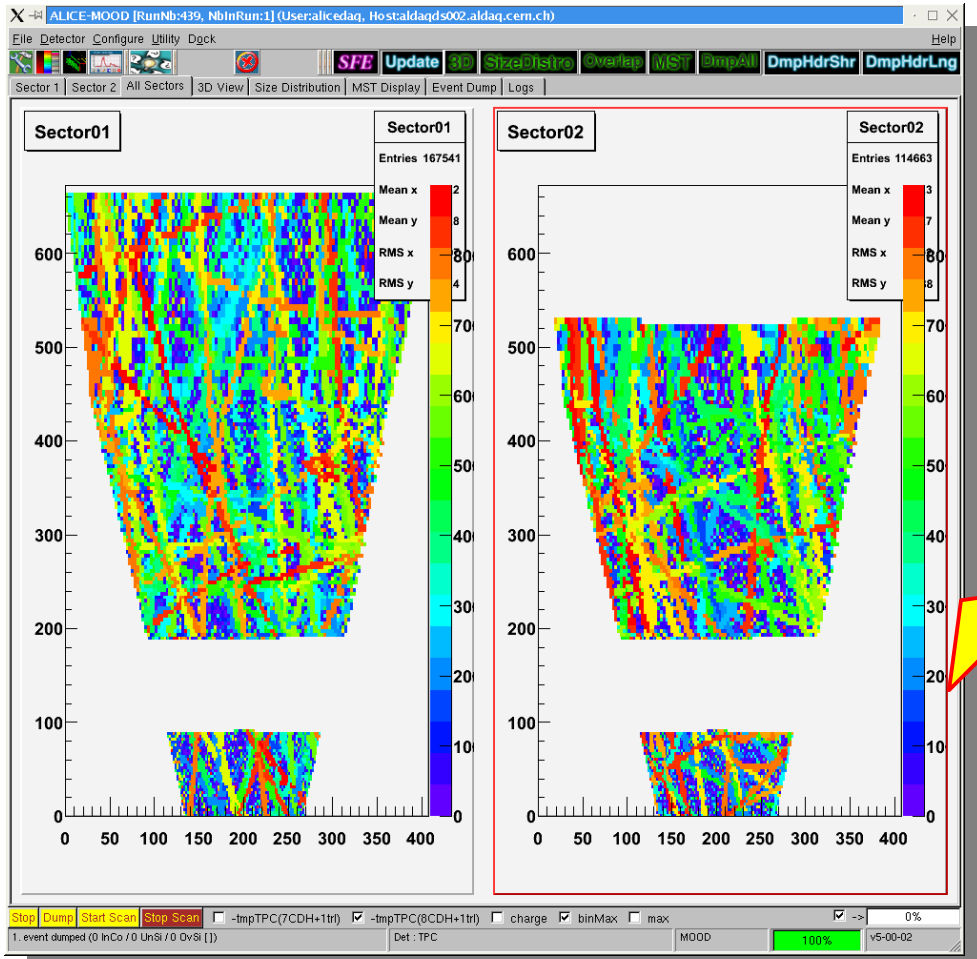



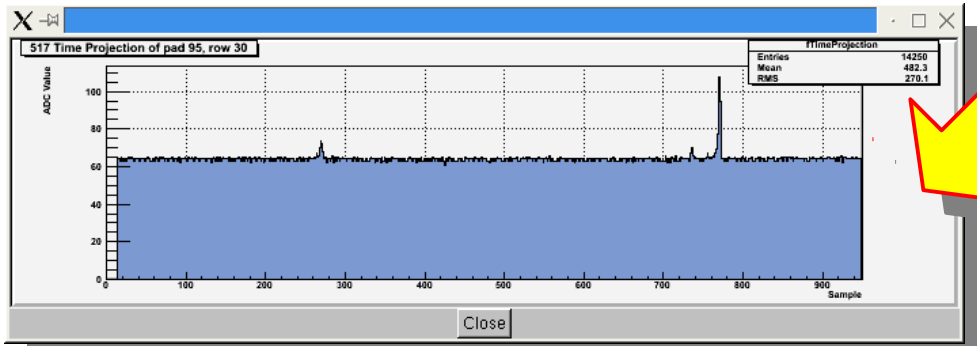



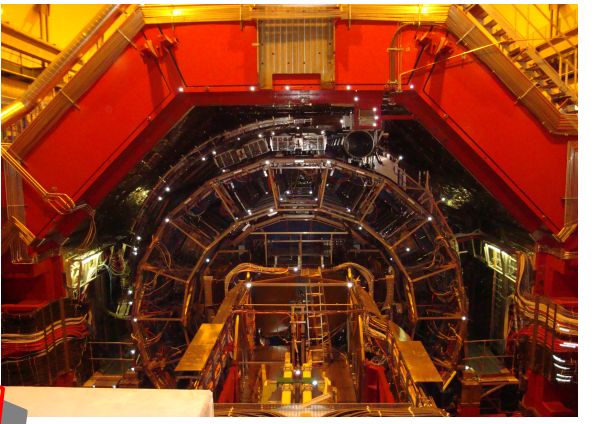

# Programlama mı Tasarım mı ?

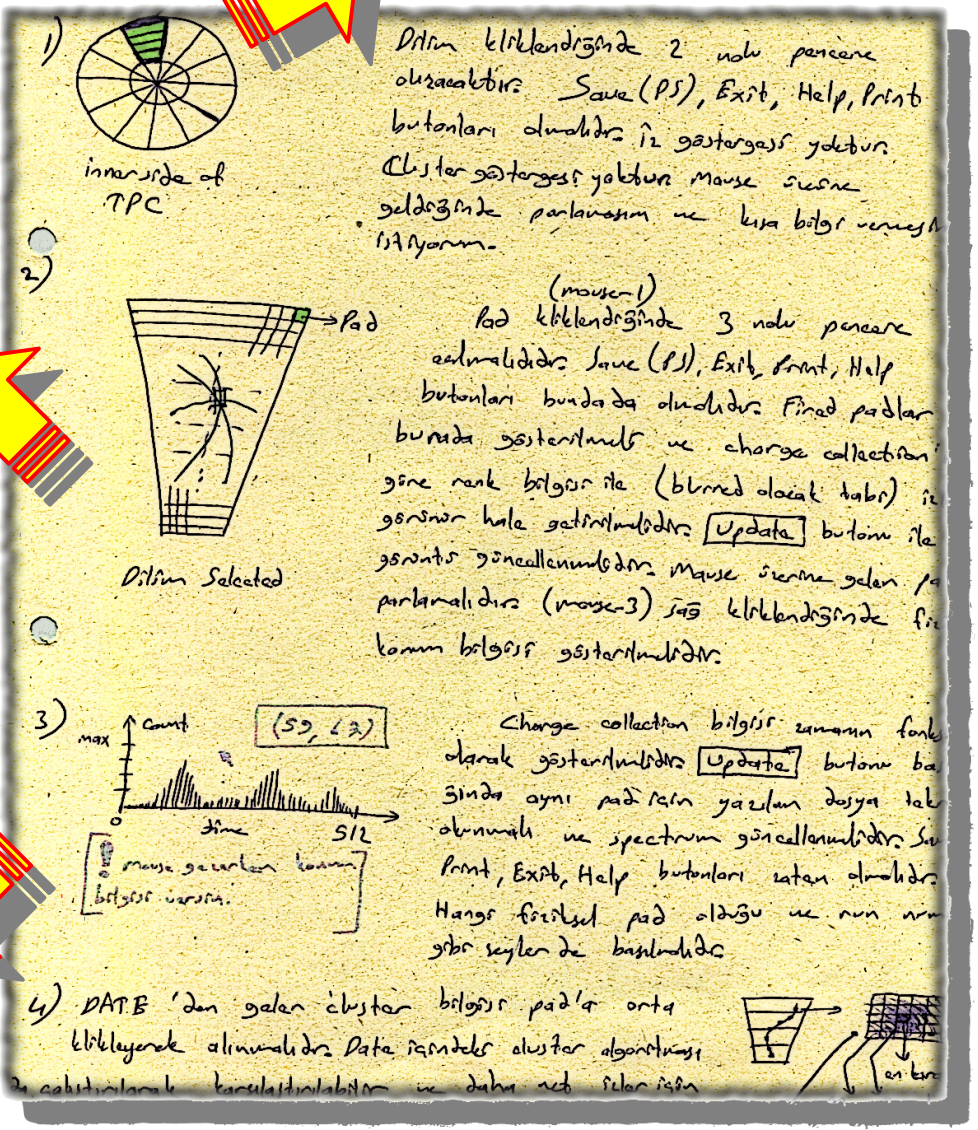
## Düşün, tasarla, yaz, uygula ve sonra programlamaya başla




2002'den bugüne





# Aksam Sefası

## Ertesi günün sabahına hazırlanması beklenen akşamlık ödevler

- **TH1F** olarak tanımlandığını gördüğünüz **BenimGuzelHistogramim** isimli bir nesne:
  - Kaynak içinde "**BenimGuzelHistogramim->ClassName()**" biçiminde kullanılmış
  - bu kaynak parçasının **ne iş** yaptığını **açıklayın**.


*



* Lolcat' ten

# ROOT – Kısa Bir Tanıtım

## ROOT'a rağmen, ROOT'a dayanarak, ROOT'la çalışmak


- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği *.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar için GUI yaratmak
- ROOT'a rağmen ROOT'la çalışmak
  - ROOT kullanım klavuzu
  - Başlangıç fikir kaynakları:
    - $ROOTSYS/tutorials dizininin etkin işlevi
    - $ROOTSYS/test dizininin etkin işlevi
  - HTML kaynak belgelendirmesi
- ROOT içinden başka kitaplıkların kullanımı
  - CERN'deki ALICE deneyinin DQM yazılımı
    - Sadeleştirilmiş veri toplama (DAQ) ilkesi
  - Algıcın ürettiği veriyi anlamak
    - Veriye erişmek ve onu anlaşılır kılmak

*