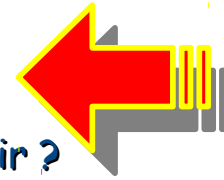


# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

- 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 ?

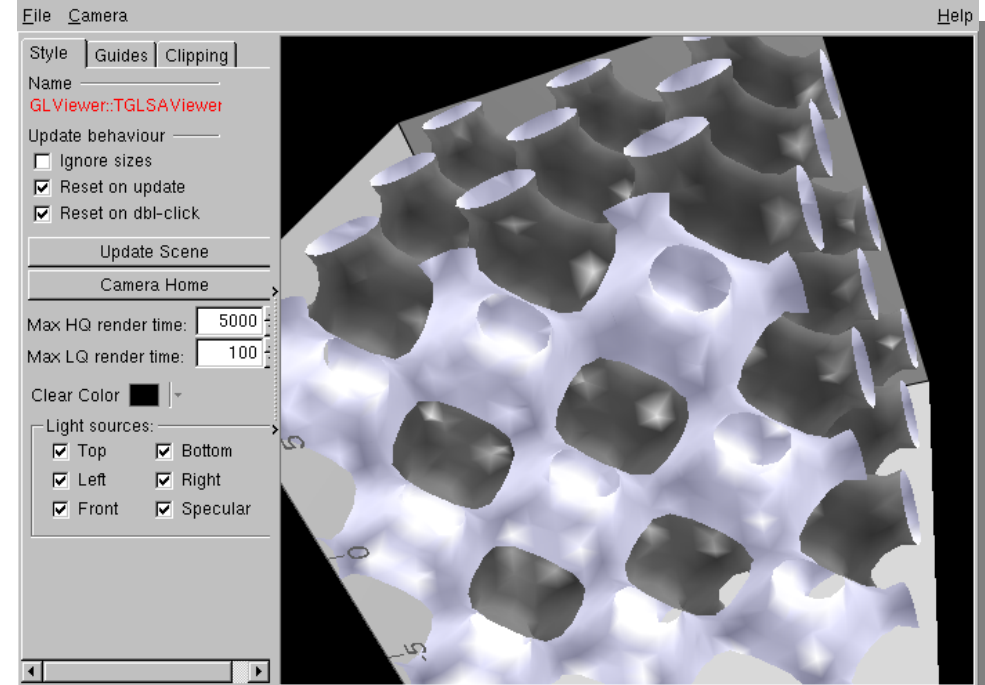
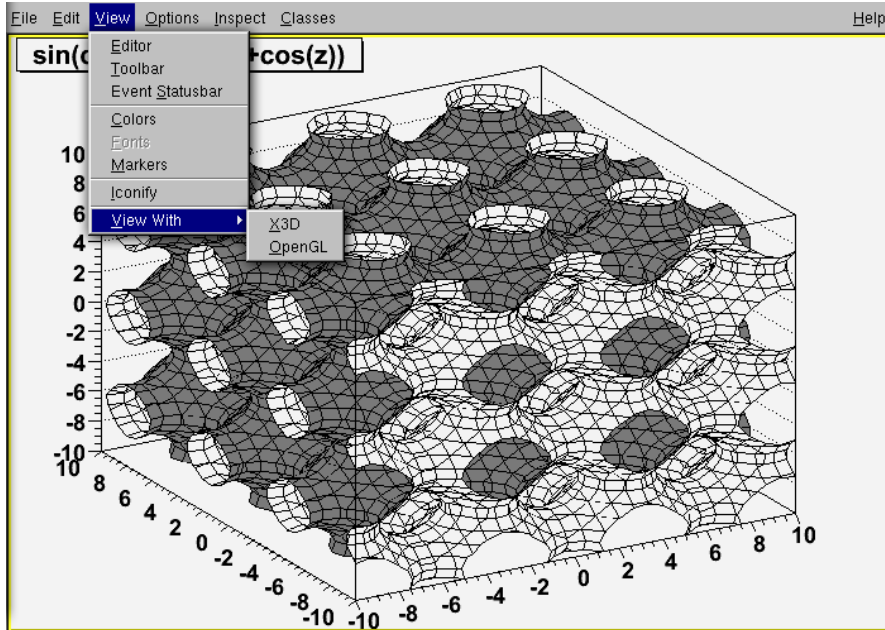
- **Bir kütüphane:** Özelleşmiş daha küçük kütüphanelerden oluşan **büyük bir class kütüphanesidir**. ROOT kütüphaneleri, geliştirilen bağımsız uygulamalarda herhangi bir kütüphaneymiş gibi kullanılabilir. Örneğin:
  - **Kullanıcı arayüzü (GUI)** geliştirme kütüphanesi olarak
  - **Signal-slot** mekanizmasını 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** çalıştıyormuş 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

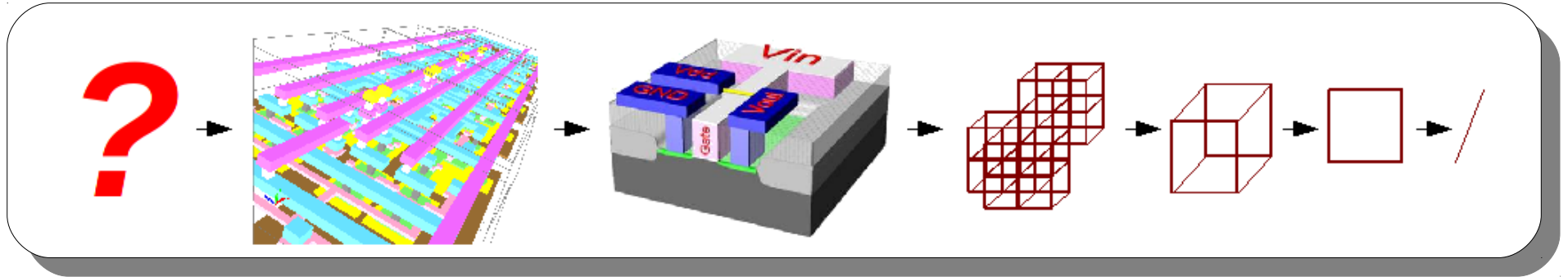
```
root [0] TF3 f1("MerabaDunya", "sin(cos(x)+sin(y)+cos(z))", -10,10,-10,10,-10,10)
root [1] f1->Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2] MerabaDunya->SetTitle("Baska bi shey !..")
```



# 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 kullanılabilirliğini** artırır
  - ➔ Class' tan class **türetme**, varolan nesnelere geliştirmeyi ve değiştirmeyi olanaklı kılar
  - ➔ Class' lar arasındaki **ast-üst ilişkisi** (hiyerarşi), **gerçek dünyadaki nesnelere** daha kolay oluşturulmasına imkan verir.



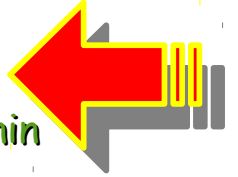
- ➔ Geliştirilen **yazılımın karmaşıklığı** düşük seviyede kalır, bilgi class' lar içinde düzenli biçimde yerleşiktir; tekdüze bir kaynak içinde küresel değişkenler (globals) üzerinde dağılmış durumda değildir.
- ➔ Yazılımdan nesnelere çıkarılması ya da yeni nesnelere eklenmesi, **genel mimariyi değiştirmez**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

# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

- 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



\*

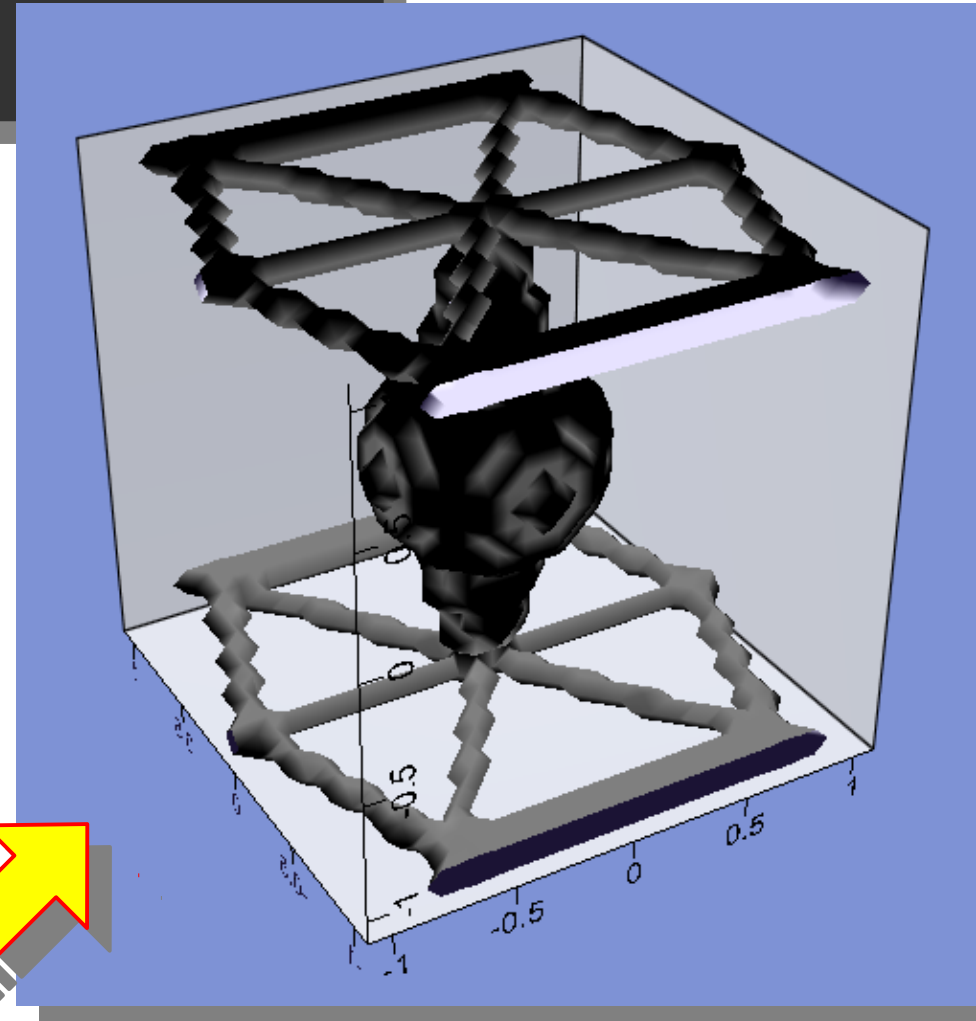
# Yükleme ve Tie Fighter Matematiği

## 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 değişkenleri 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::Abs
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()
```



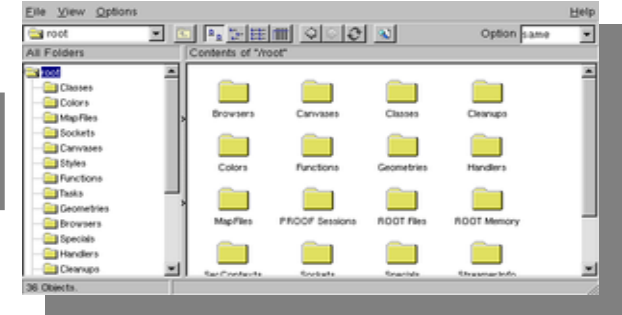
# Kullanım

## 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)

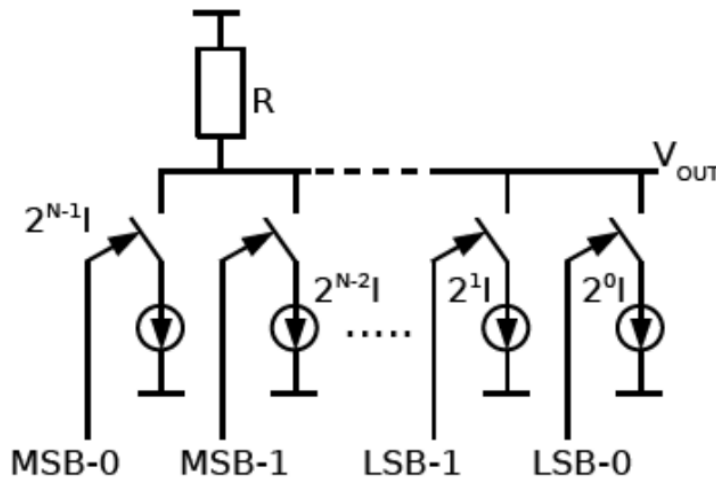
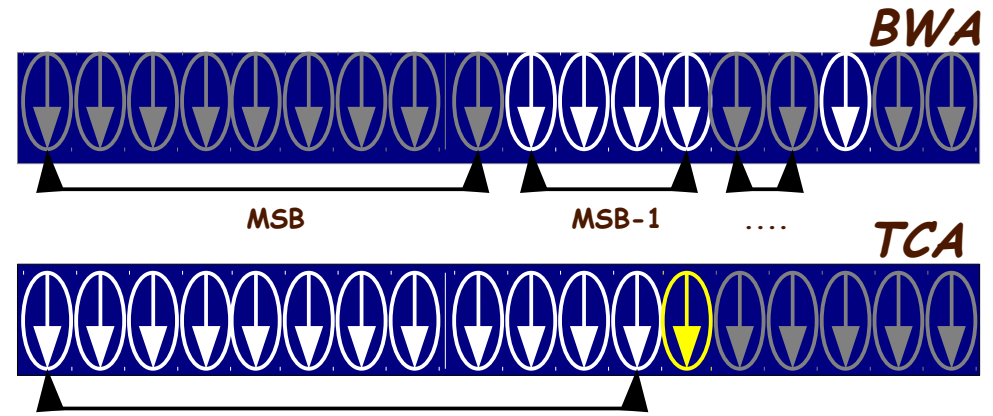
- ◆ 10-Bit D/A çevirici tasarlanacak
- ◆ İki farklı mimari var
- ◆ Hangisi daha uygun belirlemeliyiz ?
- ◆ Ölçülebilir karşılaştırma gerekli
- ◆ Monte Carlo (MC) yapılmalı

→ Çan eğrisinden,  $x_c = 1$  ve  $\sigma = 0.02$  olmak üzere **rastlantısal sayı** üret

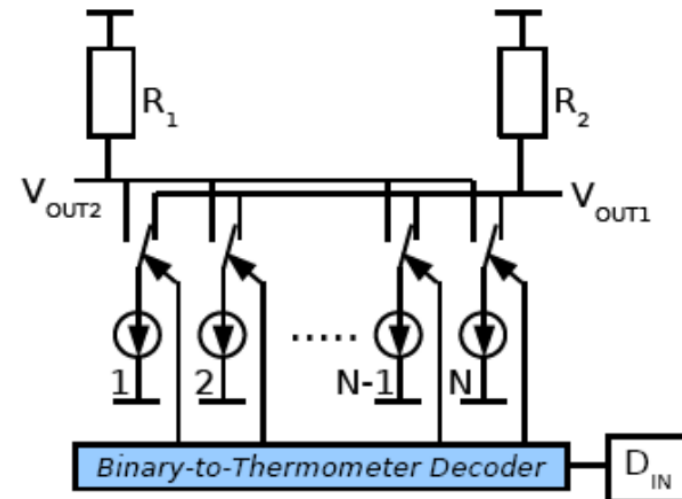
→ Bunlar her iki D/A'yı da oluşturacak birim akım kaynakları olsun:

- **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, 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-1, 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-1, 0, nop-2);
```

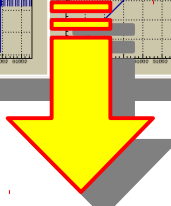
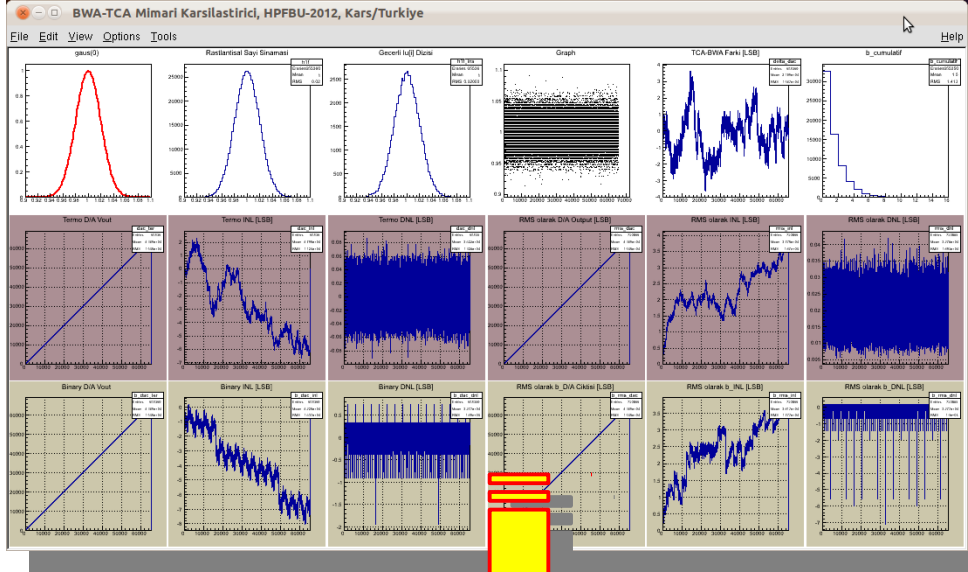
```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirici.C mimariKarsilastirici2.C
182 b_inl[i]=(1+1.0-b_dac[i]);
183 b_rms_inl->SetBinContent(i, b_rms_inl->GetBinContent(i)+b_inl[i]*b_inl[i]);
184 if (i>0 && i<nop) {
185 b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
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)/noofiteration4rms));
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->GetBinContent(i)/noofiteration4rms));
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 }
```

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

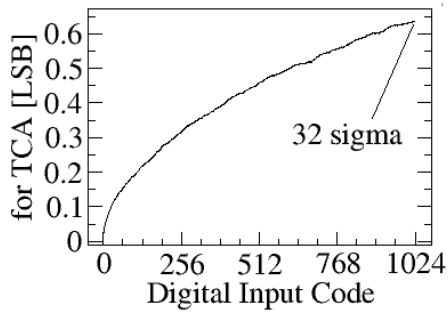
# Kullanım

## Betik yazmak (Mimari Karşılaştırmacı)

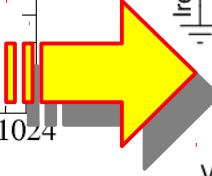
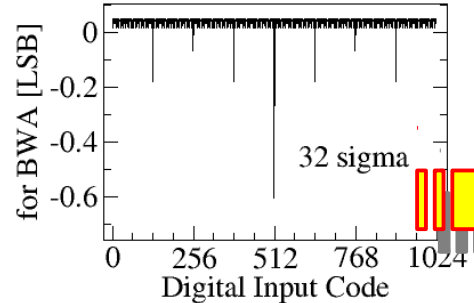
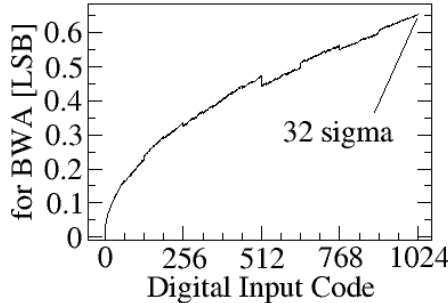
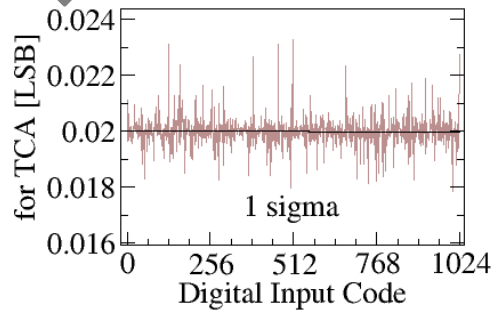
- Seçimimi yaptım:
  - INL' ler çok farklı değil
  - DNL' ler farklı, TCA' nin davranışı çok daha iyi, ama çok yer kaplayacak
- D/A' ımı tasarladım ve dedektör çalışanlara teslim ettim
- Çalıştayda sundum
  - Görev tamam !!



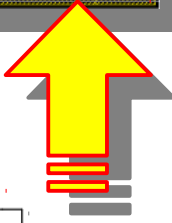
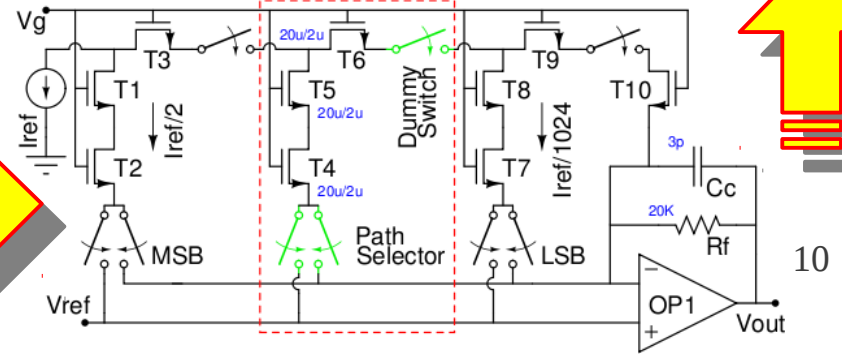
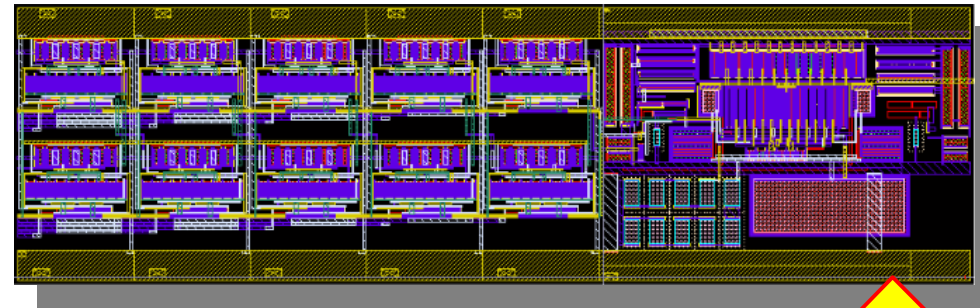
Architecture Comparison INLs



Architecture Comparison DNLs



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



# Kullanım

## Betiği \*.so kitaplığına derlemek

- Mimari karşılaştırmacı 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 çizgi 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 kütüğü 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 x mimariKarsilastirici2.C x
27 #include <TFile.h>
28 #include <TNTuple.h>
29 #include <TH2.h>
30 #include <TProfile.h>
31 #include <TCanvas.h>
32 #include <TFrame.h>
33 #include <TROOT.h>
34 #include <TSystem.h>
35 #include <TRandom.h>
36 #include <TBenchmark.h>
37 #include <T Cint.h>
38 #include <TGraph.h>
39 #include <TF1.h>
40
41 int mimariKarsilastirici2() {
42     gROOT->Reset();
43
44     // edit these parameters for other simulations -----
45     int noofbits=10;
46     int noofiteration4rms=100, nob=100; // nob : no of bins in distro
47     double sigma=0.6, centroid=1.0;
48     const Int_t KUPD = 50;
49     bool update=KTRUE;
50     //-----
51
52     int nop=pow(2.0, noofbits);
53     int boyut = pow(2, 1*noofbits);
54     double Iu[boyut]; // 2*noofbits current source
55     int lastincremented=0;
56     TCanvas *c1 = new TCanvas("c1","Binary vs Thermometer",1200,850);
57     TPad *pad1 = new TPad("pad1","Unit Current Source",0.0,0.0,1.0,1.0);
58     pad1->Draw();
59     pad1->Divide(5, 4, 0.001, 0.001);
60     pad1->cd(5)->SetFillColor(47); pad1->cd(5)->SetGrid();
61     pad1->cd(6)->SetFillColor(47); pad1->cd(6)->SetGrid();
62     pad1->cd(7)->SetFillColor(47); pad1->cd(7)->SetGrid();
63     pad1->cd(8)->SetFillColor(47); pad1->cd(8)->SetGrid();
64     pad1->cd(9)->SetFillColor(47); pad1->cd(9)->SetGrid();
65     pad1->cd(10)->SetFillColor(47); pad1->cd(10)->SetGrid();
66     pad1->cd(11)->SetFillColor(21); pad1->cd(11)->SetGrid();
67     pad1->cd(12)->SetFillColor(21); pad1->cd(12)->SetGrid();
68     pad1->cd(13)->SetFillColor(21); pad1->cd(13)->SetGrid();
69     pad1->cd(14)->SetFillColor(21); pad1->cd(14)->SetGrid();
70     pad1->cd(15)->SetFillColor(21); pad1->cd(15)->SetGrid();
71     pad1->cd(16)->SetFillColor(21); pad1->cd(16)->SetGrid();
72
73     pad1->cd(1):
```

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirici.C x mimariKarsilastirici2.C x
198     if (i>0 && i<nop) {
199         b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
200         b_rms_dnl->SetBinContent(i-1, b_rms_dnl->GetBinContent(i-1)+(b_dnl[i-1]-Iu[0])
201     }
202     b_dac_inl->SetBinContent(i, b_inl[i]-Iu[0]);
203     if (i>0) b_dac_dnl->SetBinContent(i-1, b_dnl[i-1]);
204 }
205 pad1->cd(12);
206 b_dac_inl->Draw();
207 pad1->cd(13);
208 b_dac_dnl->Draw();
209 if (update) c1->Update();
210 pad1->cd(17);
211 delta_dac->Draw();
212 pad1->cd(18);
213 b_cumulatif->Draw();
214 pad1->cd(11);
215 b_dac_ter->Draw();
216
217 } // end loop
218
219 for (int i=0 ; i<nop ; i++) { // calculate RMS
220     rms_dac->SetBinContent(i, sqrt(rms_dac->GetBinContent(i)/noofiteration4rms));
221     if (i<nop-1) rms_dnl->SetBinContent(i, sqrt(rms_dnl->GetBinContent(i)/noofiteration4rms));
222     rms_inl->SetBinContent(i, sqrt(rms_inl->GetBinContent(i)/noofiteration4rms));
223     b_rms_dac->SetBinContent(i, sqrt(b_rms_dac->GetBinContent(i)/noofiteration4rms));
224     if (i<nop-1) b_rms_dnl->SetBinContent(i, 1.55-3*(-Iu[0]/2+sqrt(b_rms_dnl->GetBin
225     b_rms_inl->SetBinContent(i, sqrt(b_rms_inl->GetBinContent(i)/noofiteration4rms));
226 }
227 pad1->cd(8);
228 rms_dac->Draw();
229 pad1->cd(9);
230 rms_inl->Draw();
231 pad1->cd(10);
232 rms_dnl->Draw();
233 pad1->cd(14);
234 b_rms_dac->Draw();
235 pad1->cd(15);
236 b_rms_inl->Draw();
237 pad1->cd(16);
238 b_rms_dnl->Draw();
239 if (update) c1->Update();
240
241 gBenchmark->Stop("binary_vs_thermometer");
242 return 0;
243
244
```

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

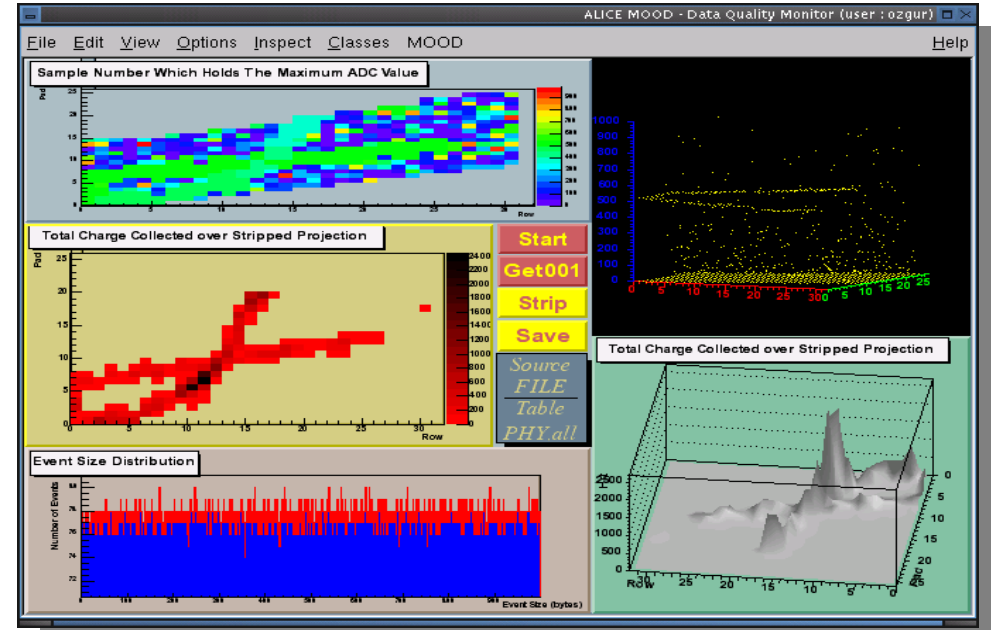
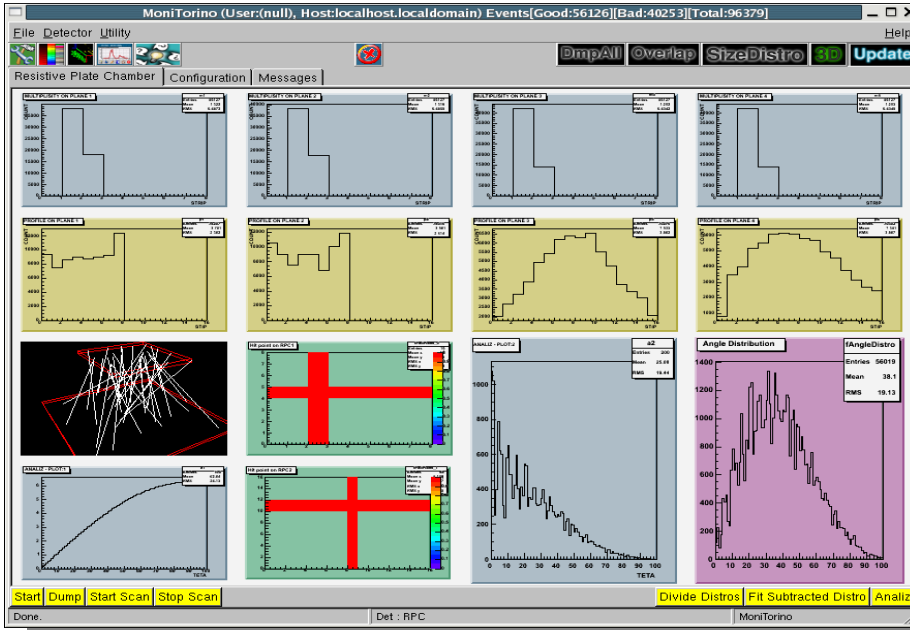
# Kullanım

## Bağımsız derlemek (uygulama yazmak)

- ◆ Mimari karşılaştırmam **\*.so** haline geldiğinde, ROOT ortamında **".L"** ile yüklenebilecek ve sanki bir **ROOT komutuymuş gibi** çalıştırılabilecektir
- ◆ Daha **yüksek hızla** çalışacaktır
- ◆ 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 x TTPLOTist.cxx x TTPLOTist.h x
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     // yonecisi 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 örneğimizde yaptığımızın aynısını yapıyoruz.
17     TF3 *islev = new TF3("islev", "sin(cos(x)+sin(y)+cos(z))", -10, 10, -10, 10, -10, 10);
18
19     // Nesnenin, kendisini görünür kılan üye işlevini çağır
20     islev->Draw();
21
22     // Kanvas nesnesinin, üzerindeki nesnelerin görünürlük
23     // durumunu yenileyen üye işlevini çağır
24     kanvas->Update();
25
26     // Uygulamayı kendi halinde çalışmaya bırak (GUI event loop' a sok)
27     benimGuzelUygulamam.Run();
28
29     // Sonlandırıldığında işletim sistemine başarı işareti göndererek çık
30     return 0;
31 }
```

→ Başlıkları ekle

→ Alışıl gelmiş C/C++ başlangıcı

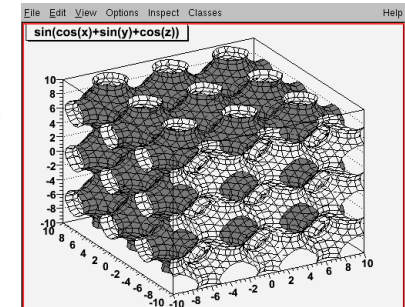
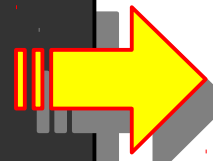
Ln 13, Col 69 INS

# 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$ _
```

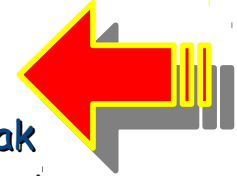


# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

- 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

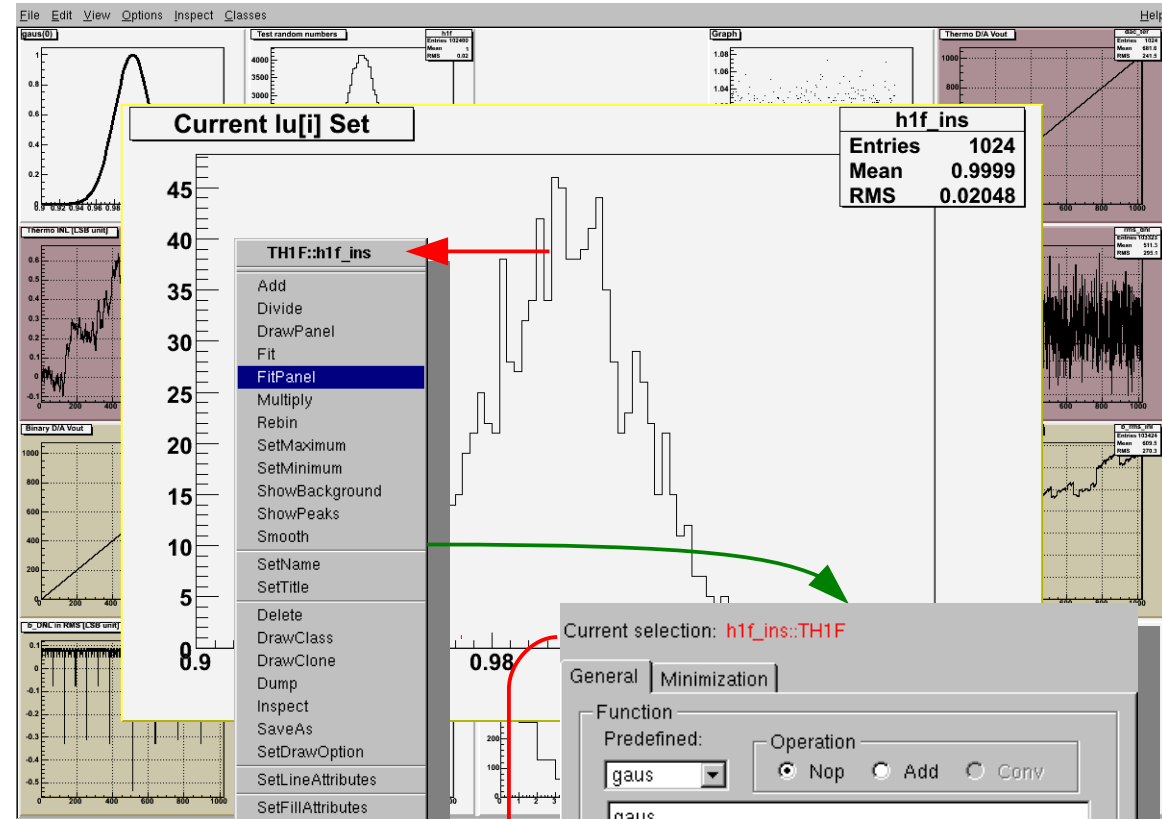
## Kullanıcı ile etkileşim



◆ Açılan tüm pencereler ve bu pencerelerde görünen **HERŞEY ama HERŞEY** ya ROOT class kitaplığından yaratılmış **NESNELERDİR** (histogram, eksen, başlık, alt-pencere v.b.) ya da ROOT class kitaplığı aracılığı ile ulaşılabilecek başka kitaplıklardan yaratılmış başka nesnelerdir (x11 kitaplığından yaratılan nesnelere ve pencere yönetici işlevleri gibi).

◆ **Sağ tıklama**, öbekte (heap) oluşturulmuş bu nesnelere **üye işlevlerinden** (class member function) bazılarının **erişim sağlar**.

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



Current selection: h1f\_ins::TH1F

General | Minimization

Function

Predefined: Operation

gaus

Selected: gaus

Set Parameters...

Fit Settings

Method: Chi-square

Linear fit:

Robust: 1.00

No Chi-square:

Fit Options

Integral:

Best errors:

All weights = 1:

Empty bins, weights=1:

Use range:

Improve fit results:

Add to list:

Draw Options

SAME:

No drawing:

Do not store/draw:

Advanced...

Eit Reset Close

LIB Minuit MIGRAD ltr: 5000 Prn: DEF

Name	Fix	Bound	Value	Min	Set Range	Max	Step	Errors
Constant	<input type="checkbox"/>	<input type="checkbox"/>	1	-3		3	0.3	-
Mean	<input type="checkbox"/>	<input type="checkbox"/>	1	-3		3	0.3	-
Sigma	<input type="checkbox"/>	<input type="checkbox"/>	0.02	-0.06		0.06	0.006	-

Immediate preview

Reset Apply OK Cancel

# Kullanım

## Kullanıcı ile etkileşim

Style Name: stats::TPaveStats

Line: 1

Fill: 1

Text: 6. helvetica bold, 12 Middle, Left

Stat Options: Name, Entries, Overflow, Mean, Underflow, RMS, Skewness, Integral, Kurtosis, Errors

Fit Options: Values, Probability, Errors, Chi

Style Name: stats::TPaveStats

Line: 1

Fill: 1

Text: 6. helvetica bold, 12 Middle, Left

Stat Options: Name, Entries, Overflow, Mean, Underflow, RMS, Skewness, Integral, Kurtosis, Errors

Fit Options: Values, Probability, Errors, Chi

Style Name: fitFunc::TF1

Line: 6

Function: gaus

Update Npar: 3

X-Range: 0.9000, 1.1000

Marker: 1.0

Style Name: fitFunc::TF1

Line: 3

Function: gaus

Update Npar: 3

X-Range: 0.9000, 1.1000

Marker: 1.0

Current lu[i] Set

h1f\_ins

Entries	1024
Mean	0.9999
RMS	0.02048

TF1::fitFunc

- DrawDerivative
- DrawIntegral
- SetMaximum
- SetMinimum
- SetNpx
- SetRange
- SetParameters
- SetParNames
- SetName
- SetTitle
- Delete
- DrawClass
- DrawClone
- Dump
- Inspect
- SaveAs
- SetDrawOption
- SetLineAttributes
- SetFillAttributes
- SetMarkerAttributes

Current selection: h1f\_ins:TH1F

General Minimization

Function

Predefined: Operation:  Nop  Add  Copy

Function: gaus

Selected: gaus

Set Parameters...

Fit Settings

Method: Chi-square

Lineal fit:

Robust: 1.00

No Chi-square:

Fit Options

Integral:  Use range:

Best errors:  Improve fit results:

All weights = 1:  Add to list:

Empty bins, weights=1:

Draw Options

SAME:  No drawing:  Do not store/draw:

Advanced...

Eit Reset Close

- 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 **çalışmayı saklamaya** geldi sıra = **kaydetmek** !
- ◆ 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.

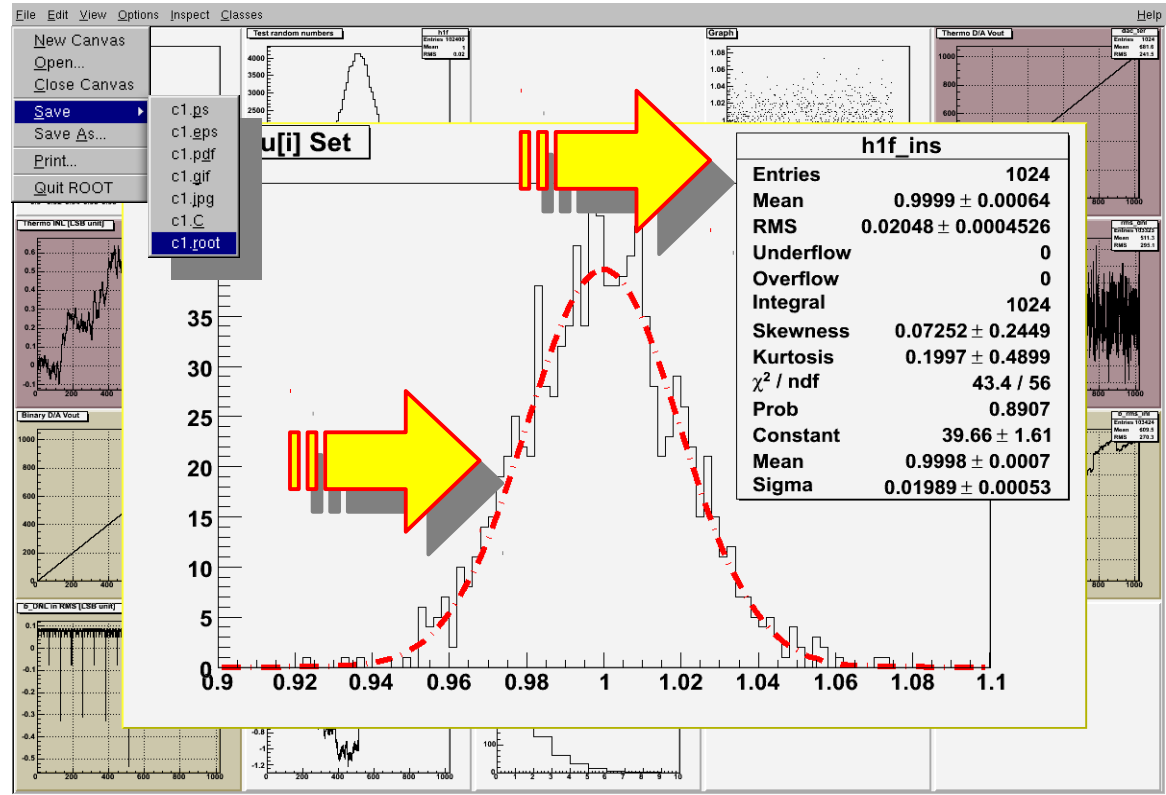
"TBrowser a")

- **Resim olarak** da saklanabilir:

→ ps, eps, gif v.b.

- Uyum eğrisi **daha kalın, kesikli ve kırmızı**

- Histogramla ilgili daha fazla **istatistik bilgi** görülüyor

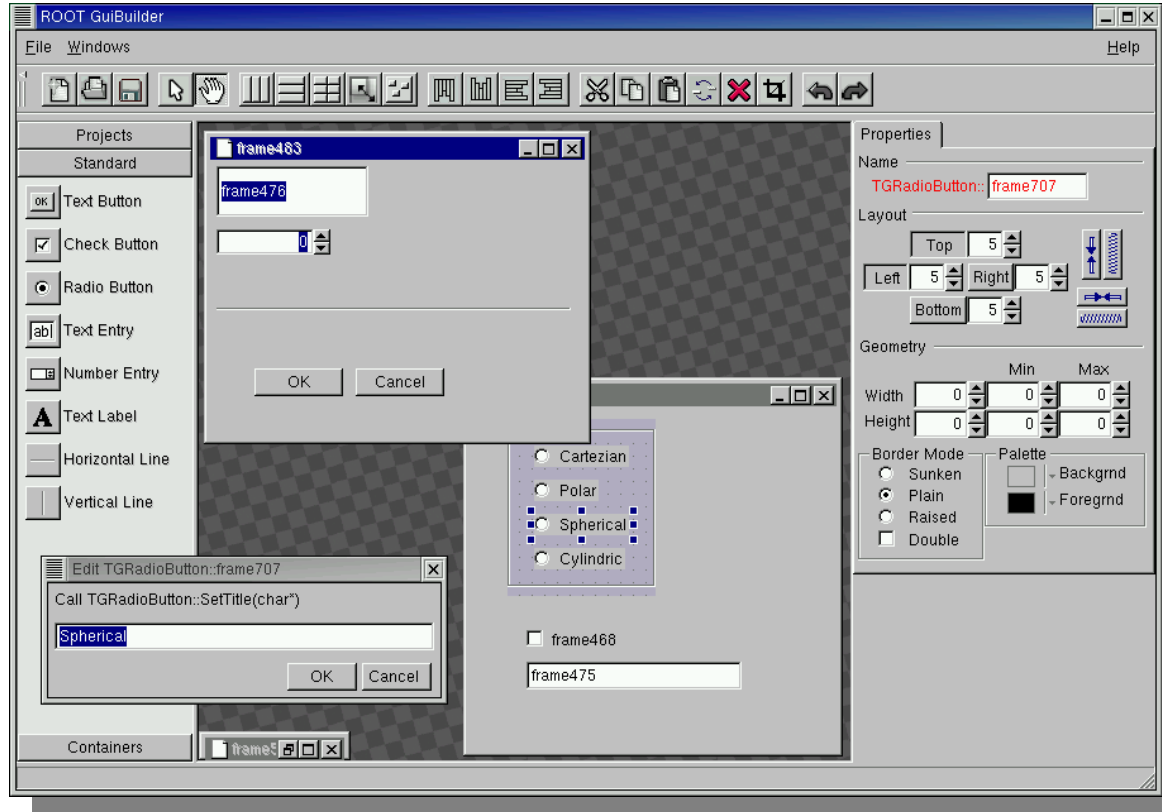


# Örnek

## 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 !!**

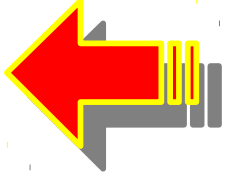
→ Zaman kazandırıcı: ROOT GUI class' larını **ezbere bilme** gerekliliği yok

→ **Sık sık kaydet, arada gözebilir !**

# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

- 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 çalışıp 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 çalışıp kendi uygulamalarınızı yazmaya başlamanız



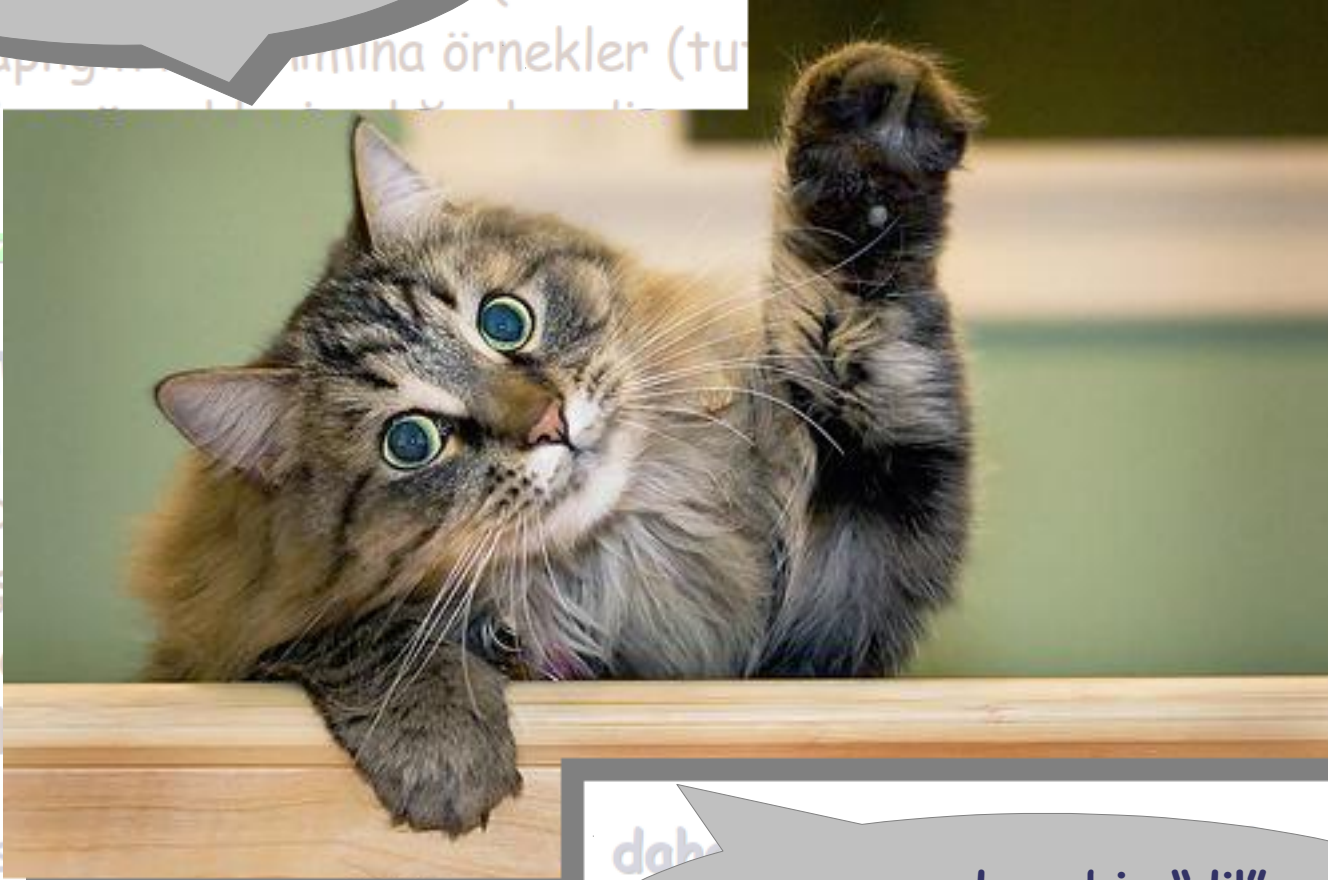
Pekiye, bu zaman kaybı neden ?

- ◆ Şanslı
- ◆ Kullanım kılavuzunu, kurulumunu, yapılandırmasını, testlerini, örneklerini, v.b. yapın. T'le çalışmak zor değil !!
- \* Hızlı değişmez, duragandır.
- 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

ROOT kullanım  
kılavuzu ~500  
sayfadır ! Üstelik ...



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



# 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 öğrenmektir**.
- ◆ Dil **konusularak** ve/veya **kullanılarak** öğrenilir.
- ◆ Bu dizin, birlikte ROOT' ça **konuşmaya başlanacak arkadaşlarımızın 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 gerek**miyor**.
  - Sadece **"tutorial" da bu işi yapan bir alıştırma vardı, oradan başlayabilirim...** diyebilmeniz gerekli

# ROOT'a Rağmen ROOT'la Çalışmak

\$ROOTSYS/test dizininin etkin işlevi

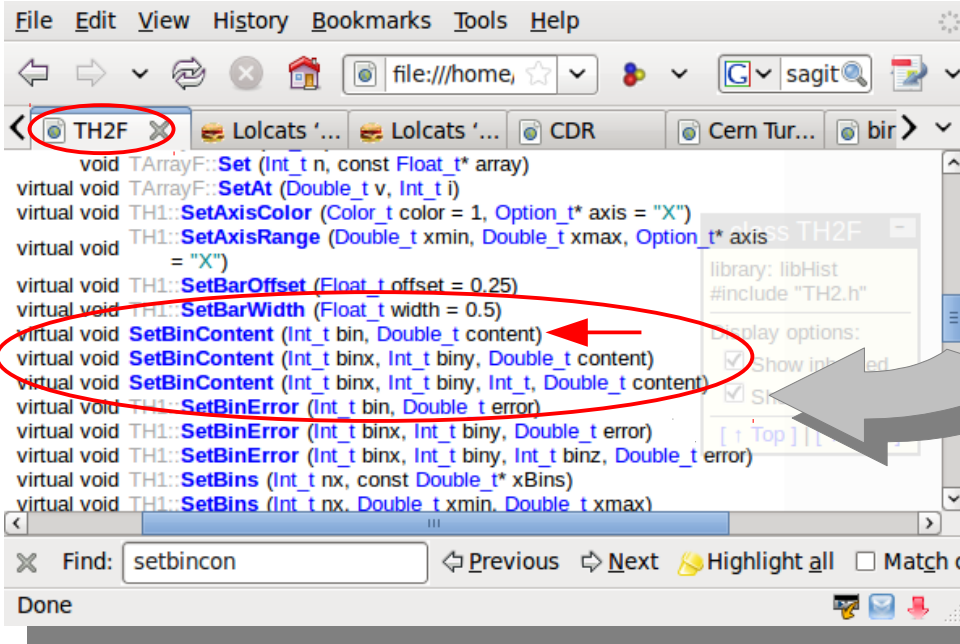
- ◆ **Uygulama** yazarken **ölümle kalım** arasındaki çizgi
- ◆ Öğrenilecek/çalışılacak bir şey **değil**, ROOT' arken devamlı olarak başvurulacak en düşük seviyeli (asıl olana en yakın mesafede) öğretmen
- ◆ **Yeni bir kitaplık kullanmak**, özellikle C/C++ gibi kitaplık desteği olmadan herşeyin zor olduğu dillerde, **yeni bir dil öğrenmektir**.
- ◆ Dil **konusularak** ve/veya **kullanılarak** öğrenilir.
- ◆ Bu dizin, birlikte ROOT' ça **konuşmaya başlanacak arkadaşlarımızın 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 **gerekmiyor**.
  - Sadece **"test' te bu işi yapan bir alıştırma vardı, oradan başlayabilirim<sub>26</sub>."** diyebilmeniz gerekli

# ROOT'a Rağmen ROOT'la Çalışmak

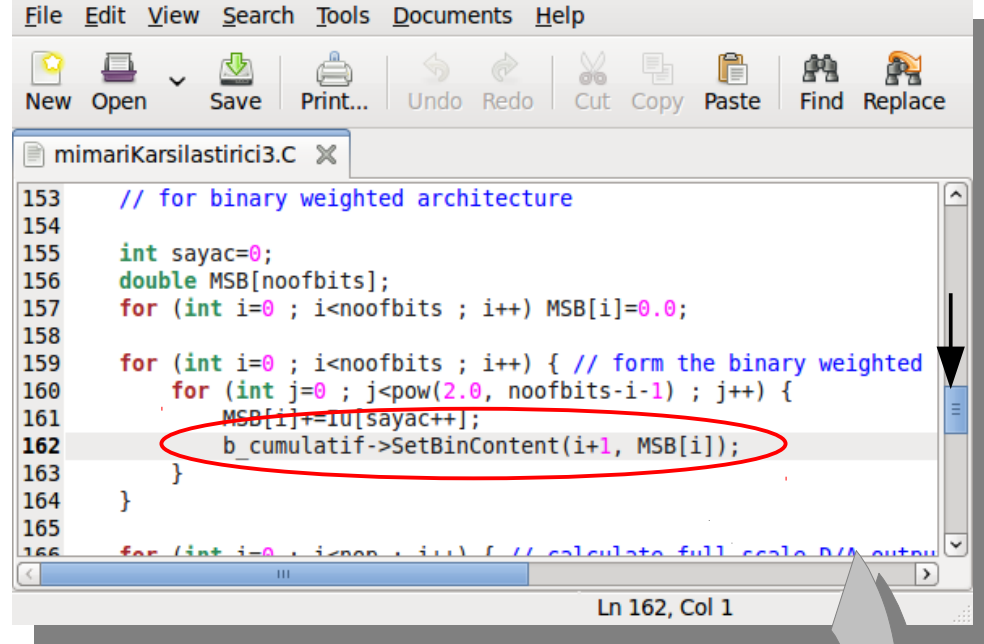
## HTML belgelendirmesi

- **Ölümlü 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 okumakla eşdeğer**, **hataya en az açık yöntem**

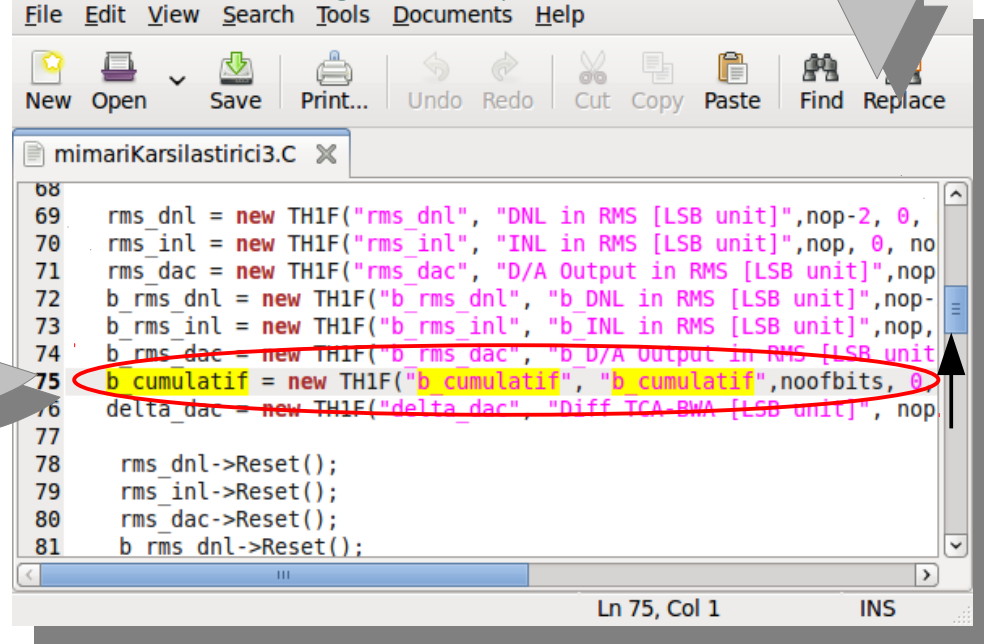
## En sevdiğiniz html yorumcu (firefox, google-chrome)



## En sevdiğiniz metin düzenleyici (gedit)



## Ctrl-F ile "b\_cumulatif" arıyor ve "ne" olduğuna bakılıyor: TH1F nesnesi



# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

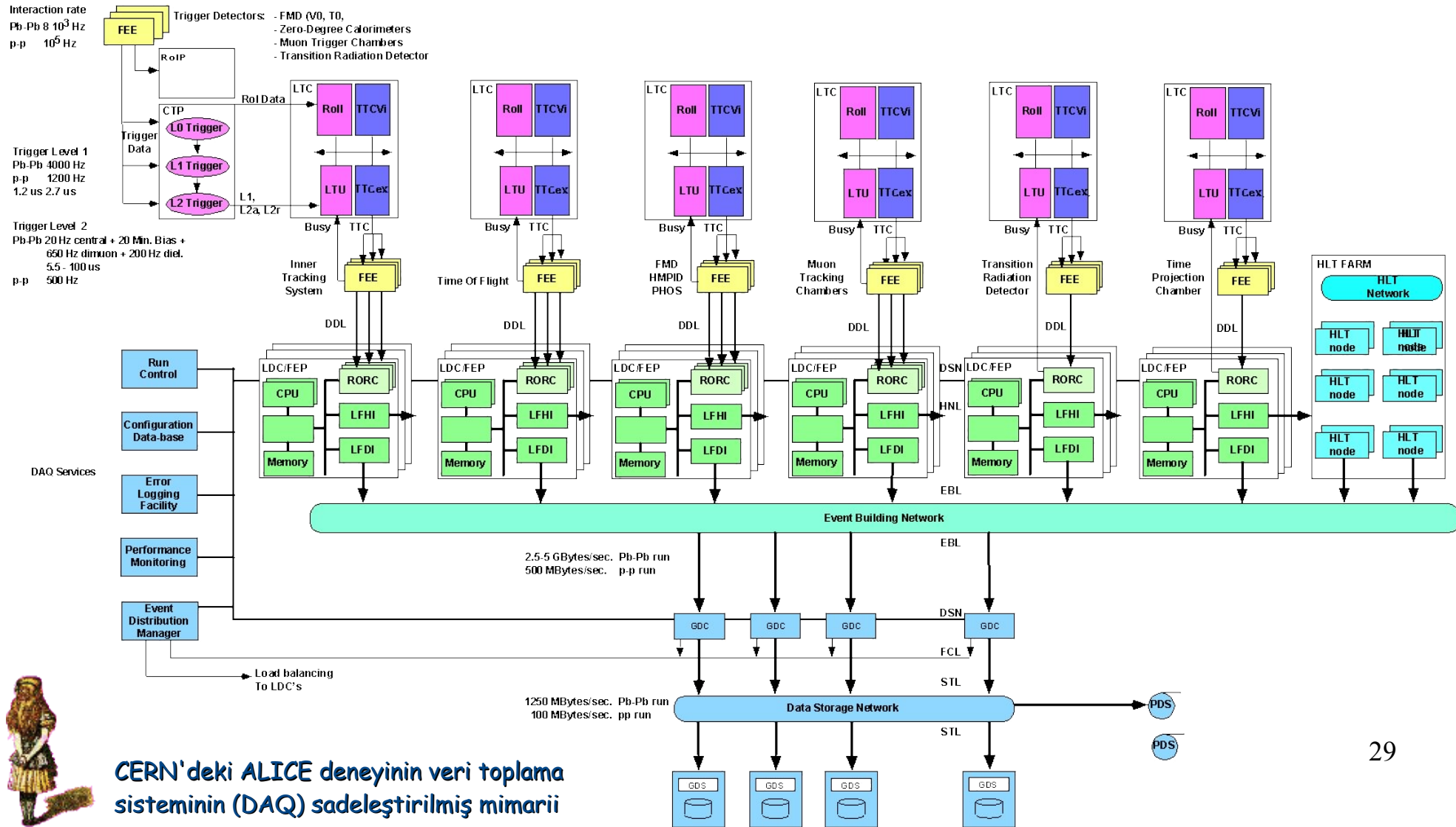
- Giriş
  - Nedir ?
  - Neden iyidir ?
- Kullanım
  - Komut verme ve betik yazma
    - Akım kipindeki iki D/A mimariinin karşılaştırılması
  - Derleme
    - Betiği \*.so kitaplığına derlemek
    - Uygulama geliştirme (bağımsız derlemek)
- ROOT kullanıcı arayüzü (GUI)
  - Kullanıcı ile etkileşim
  - Uygulamalar 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++ kitablığı** aracılığı ile sağlanır

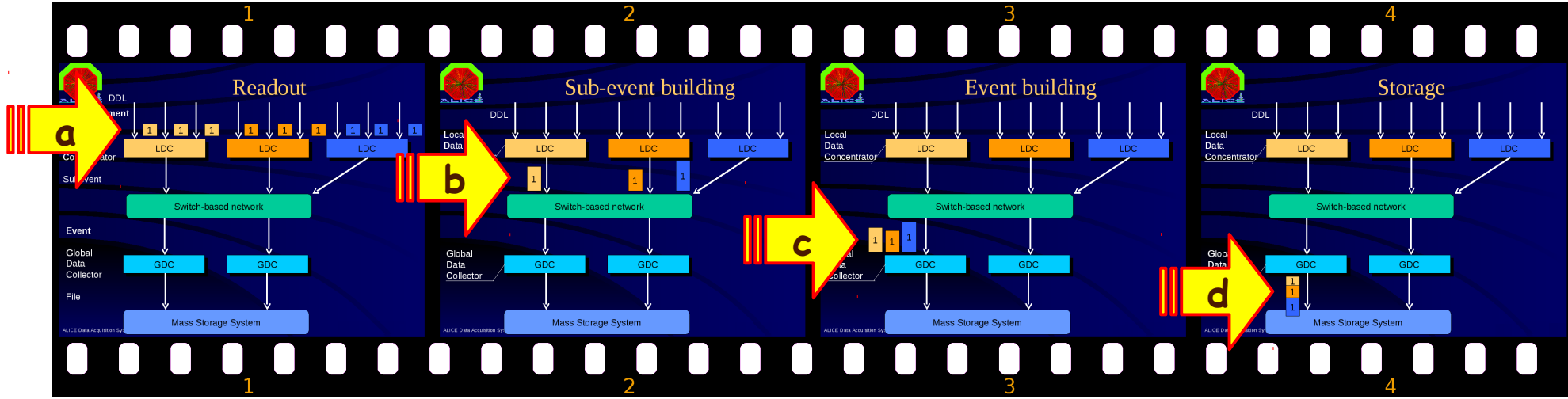


CERN'deki ALICE deneyinin veri toplama sisteminin (DAQ) sadeleştirilmiş mimarisi



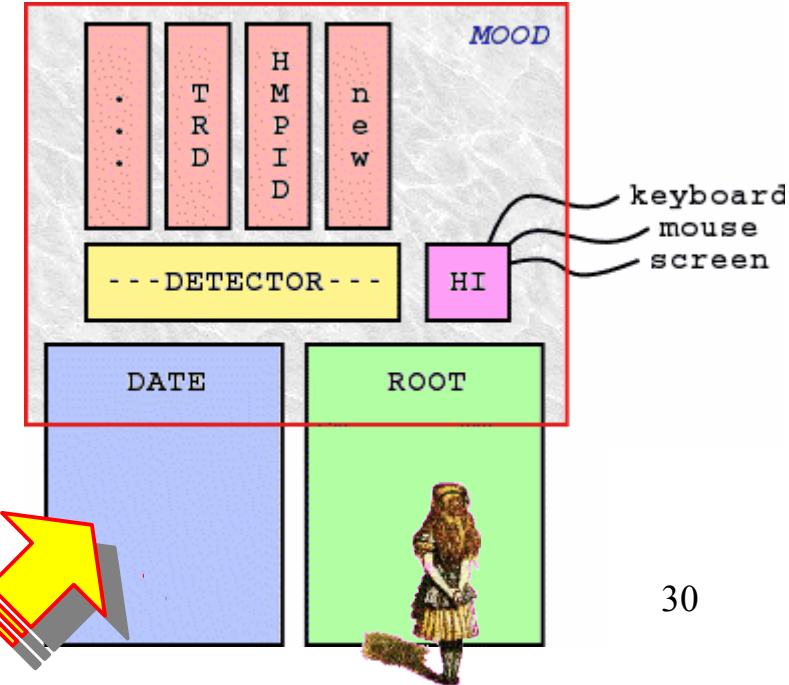
# Başka Kitaplıkların ROOT ile Kullanımı

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 üzere gönderiliyorlar **(d)**
- ➔ **Görevimiz**, veriye erişim için **DATE**, görselleştirme ve etileşimleştirme gibi diğer herşey için **ROOT** kitaplıklarını 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("${DATE_MONITOR_DIR}/{DATE_SYS}/libmonitor.so");
root [1] monitorSetDataSource(""); /* Yerel canlı izlemeyi aç */
root [2] int *event;
root [3] monitorGetEventDynamic(&event);
root [4] printf("%08x %08x \n", event[0], event[1]);
0000878c da1e5afe
Root [5] .q
oc@olmak-x200:~$ _
```

## 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 >
- ...

## Özet:

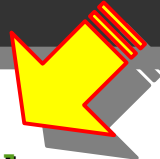
- Root [0] → İzleme kitaplığını yükle
- Root [1] → Veri kaynağını seç
- Root [2] → Verinin göçertileceği değişken
- Root [3] → Veriyi değişkene göçert
- Root [4] → İlk iki "kelime"yi göster
- Root [5] → Oturumu kapat

Büyük olay başlığı  
(super event header)

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

Alt-algıç verisi (yük, payload)

Bir başka yük

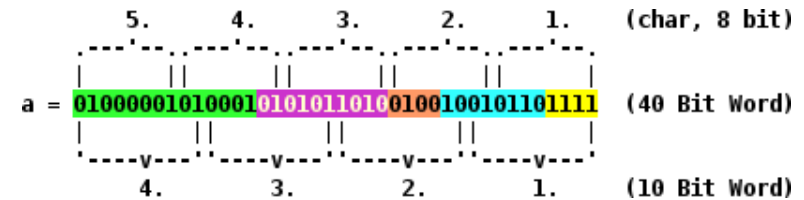


# Başka Kitaplıkların ROOT ile Kullanımı

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)

0000:0000	8c870000	fe5a1eda	44000000	04000300	07000000	78130000	00000000	f842bf06
0000:0020	01000000	00000000	00000000	f0000000	00000000	20000000	03000000	ffffff
0000:0040	d30e0b43	48870000	11000000	69000000	00000000	00000000	00000000	04000000
0000:0060	01000000	02000000	03000000	04000000	05000000	06000000	07000000	08000000
0000:0080	3ef4d043	0f3df4e0	430f3df8	d0430f3d	f8e0430f	3df4d043	0f3ef4d0	430f3ef4
0000:00a0	d0430f3d	f4d0830f	3ef4d043	0f3ef8d0	430f3df4	d0430f3d	f8d0830f	3df8e003
0000:00c0	0f3ef8d0	430f3df4	d0430f3d	f4e0830f	3ef4d083	0f3df0d0	830f3df4	d0830f3d
0000:00e0	f4e0430f	3ef4d043	0f3ef8d0	430f3df4	e0830f3d	f0d0430f	3df4e043	0f7298a1
0000:0100	aaaa00a8	66a8aa3d	f8d0830f	3ef4d043	0f3ef4d0	430f3df4	e0830f3e	f4d0430f
0000:0120	3df0d043	0f3ef8d0	430f3df8	e0430f3d	f4e0430f	3df4e043	0f3df8e0	430f3df4
0000:0140	d0830f3e	f4d0830f	3ef4e083	0f3ef8d0	430f3ef4	e0430f3d	f4d0430f	3df4d043
0000:0160	0f3df4e0	430f3df4	d0830f3d	f4d0430f	3df4e043	0f3ef8c0	430f3df4	e0430f3c
0000:0180	f8e0830f	7298a1aa	aa01a866	a8aa2bac	b0c20a2a	b0c0c20a	2bacb0c2	0a2bb0b0
0000:01a0	c20a2aac	b0c20a2b	acb0c20a	2ab0b082	0a2aacb0	c20a2bb0	b0820a2b	acb0c20a
0000:01c0	2bacb0c2	0a2bb0c0	c20a2bac	b0820a2b	b0b0c20a	2bb0b0c2	0a2bacc0	020b2bac
0000:01e0	b0c20a2b	acb0c20a	2aacb0c2	0a2bacb0	c20a2bac	b0020b2b	acb0c20a	2bacb0c2
0000:0200	0a2bacc0	c20a2cac	b0c20a72	98a1aaaa	02a866a8	aa2bacb0	020b2cac	c0c20a2c
0000:0220	b0c0020b	2cb0c0c2	0a2cb0c0	c20a2cb0	c0c20a2b	acc0020b	2cb0c002	0b2bb0c0
0000:0240	020b2cac	c0c20a2c	b0c0020b	2cb0c002	0b2cb0b0	c20a2bb0	c0020b2c	acc0c20a
0000:0260	2bacc002	0b2cb0c0	c20a2cb0	c0c20a2c	acb0c20a	2bb0c002	0b2bacc0	c20a2cb0
0000:0280	c0020b2c	b0c0020b	2bacc0c2	0a2bb0c0	c20a7298	a1aaaa03	a866a8aa	28a070c2
0000:02a0	09279c60	c209279c	80020a27	9c70c209	289c70c2	09279c70	c20928a0	80c20928
0000:02c0	a070c200	72a070c2	0072a000	c200720c	70070c27	0c80c200	720c70c2	00709c70



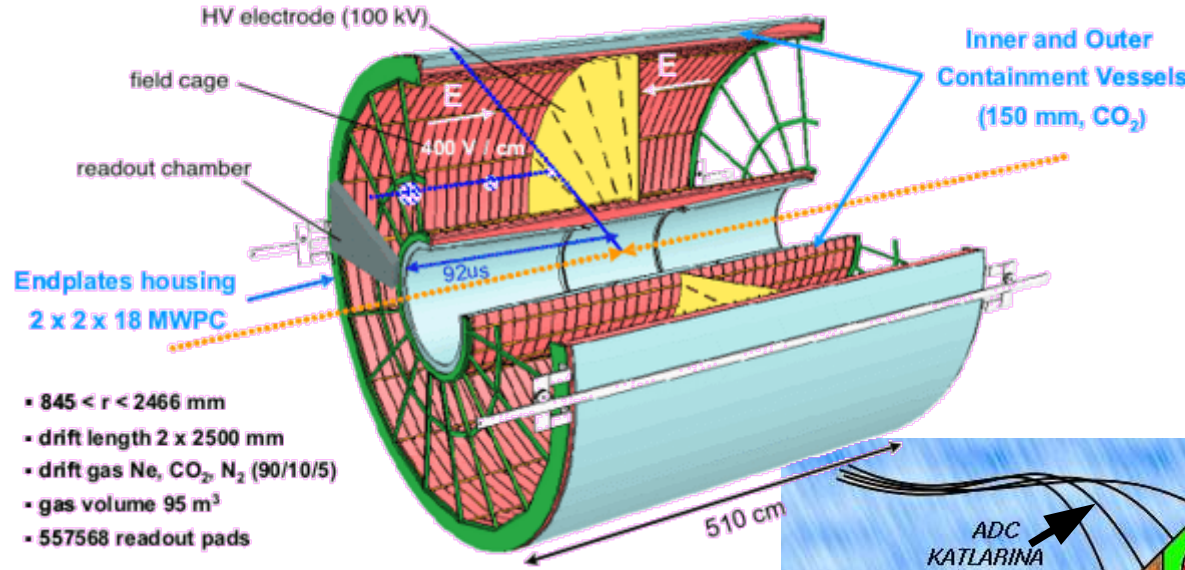
Seq.	bin	dec	(10 Bit Words, trailer)
Channel.	1111	15	(yellow)
Hardware.	10010110	150	(light blue)
Pattern A	0100	4	(orange)
10BitWord	0101011010	346	(magenta)
2AAA Pat.	01000001010001	4177	(green)



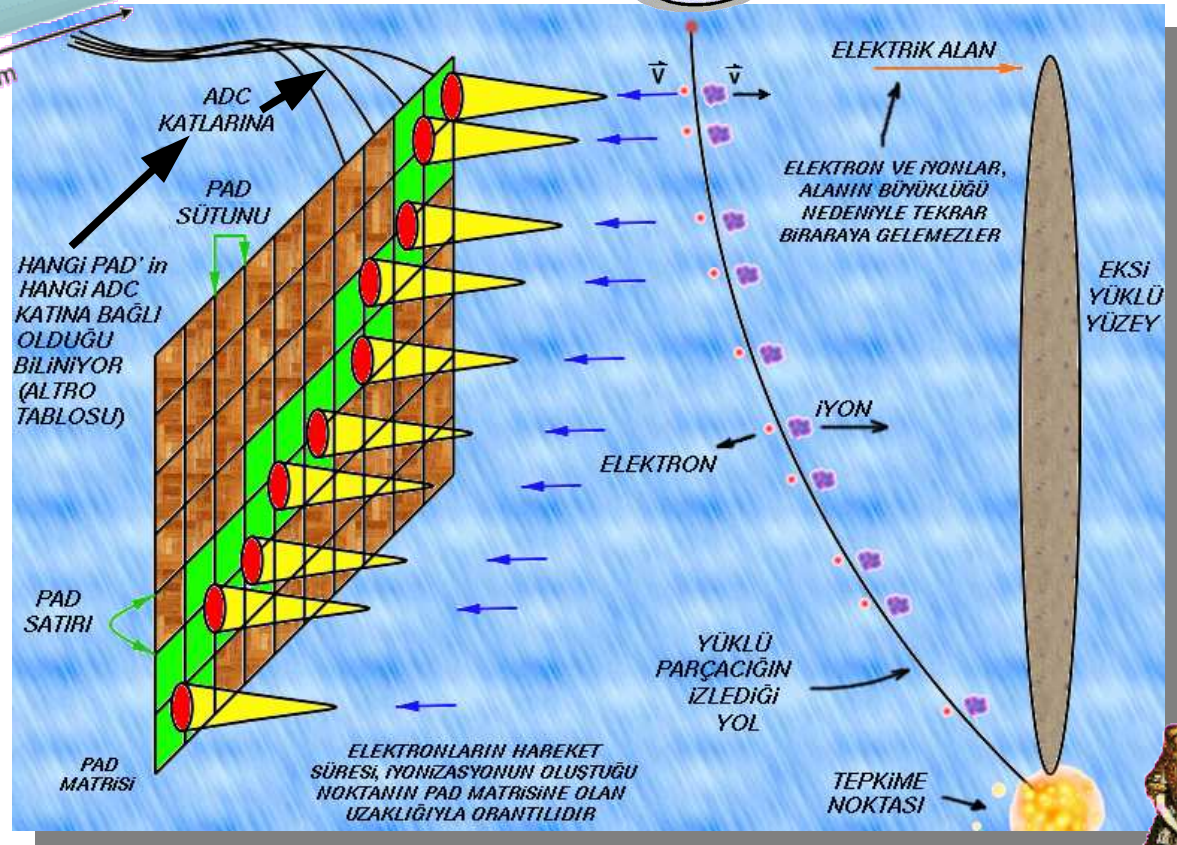
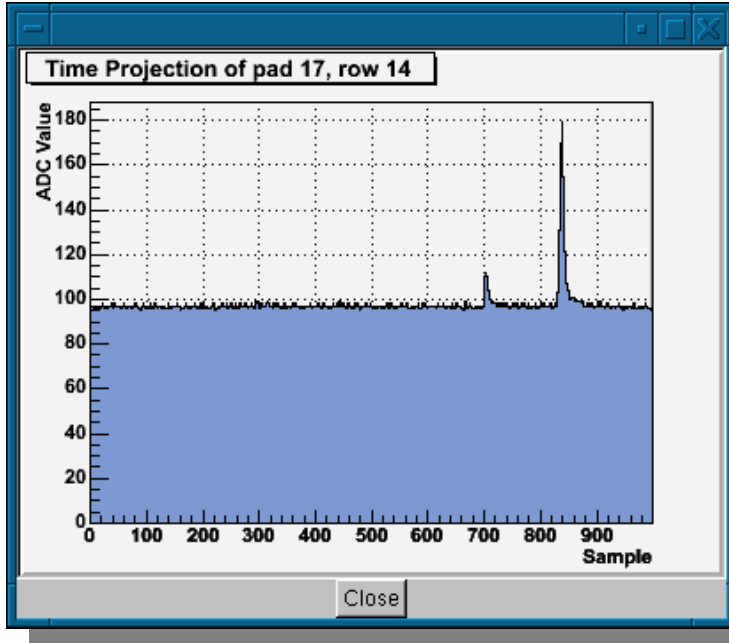
# Başka Kitaplıkların ROOT ile Kullanımı

Algıcın ürettiği 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

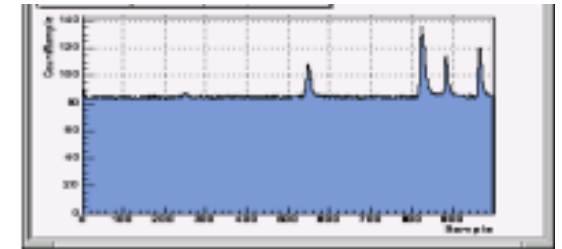
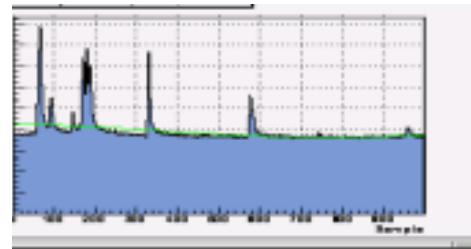
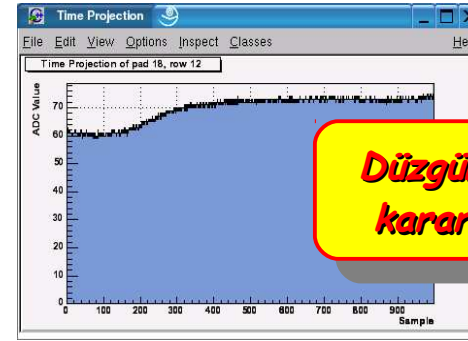
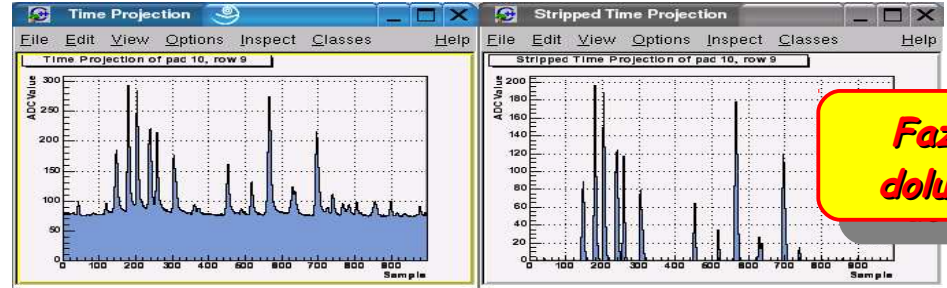
- Takılı kalma (Stack-at) hataları
- Ayarlanamaz (non configurable) FE, v.b.
- Beklenmedik davranış

## ■ Algıçla ilgili konular

- Gaz karışımı ve akışı
- Doluluk oranı (occupancy), v.b.
- Beklenmedik davranış

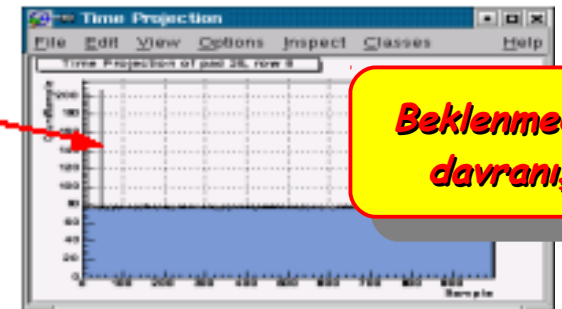
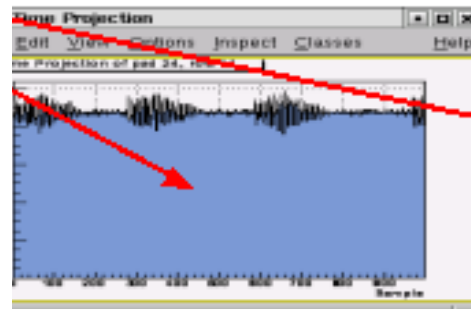
## ■ Genel

- Sıcaklık ve güç kaynağındaki dalgalanmalar
- Tepelerin biçimi (peak formation)
- Karar seviyesindeki (baseline) dalgalanmalar, v.b.
- Beklenmedik davranış



→ Hep aynı değeri üreten kanal

→ Karar seviyesindeki aşırı dalgalanma



Beklenmedik davranış

Fazla doluluk

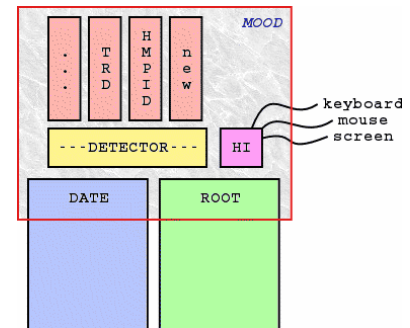
Düzensiz karar seviyesi

# 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 -lthread -lm -lmst -ldynamic -lnonStand -lshift -lcustomDet -lmnrno {...} 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 sahiptir; 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
```



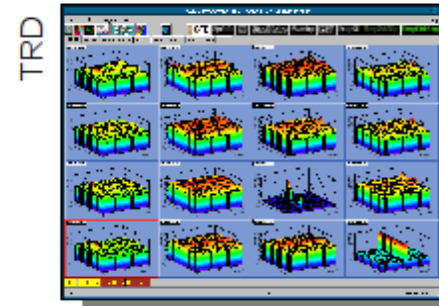
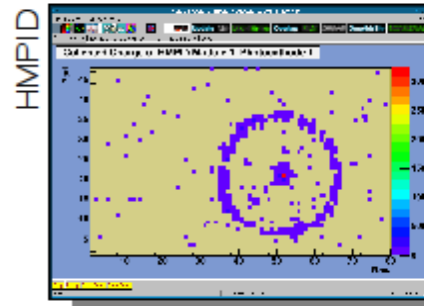
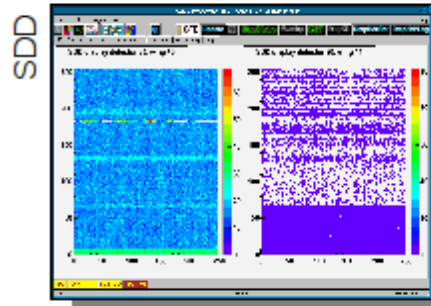
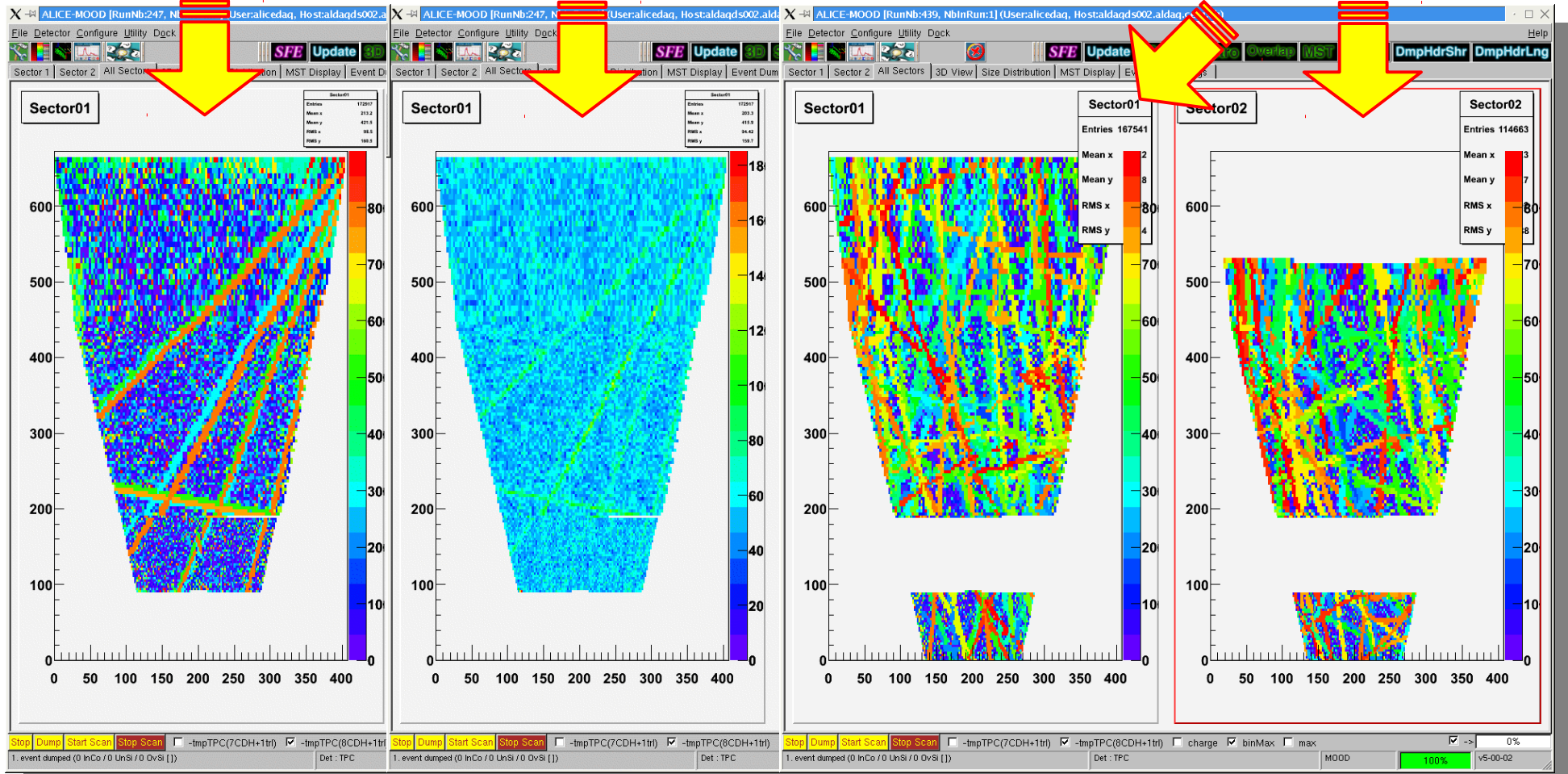
# Başka Kitaplıkların ROOT ile Kullanımı

## Kullanıma hazır uygulamanın teslimi

İki farklı lazer olayı

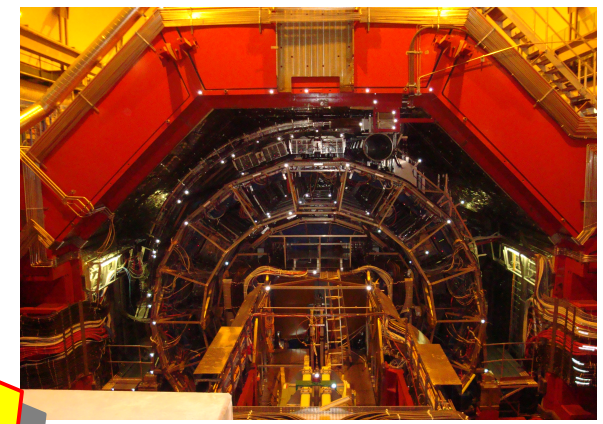
Aynı lazer olayı farklı algoritmalarla görünür kılınmış

Bir kozmik olay

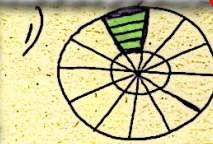
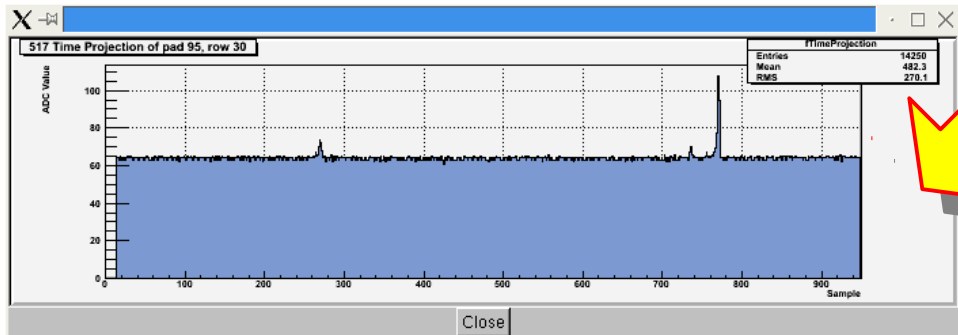
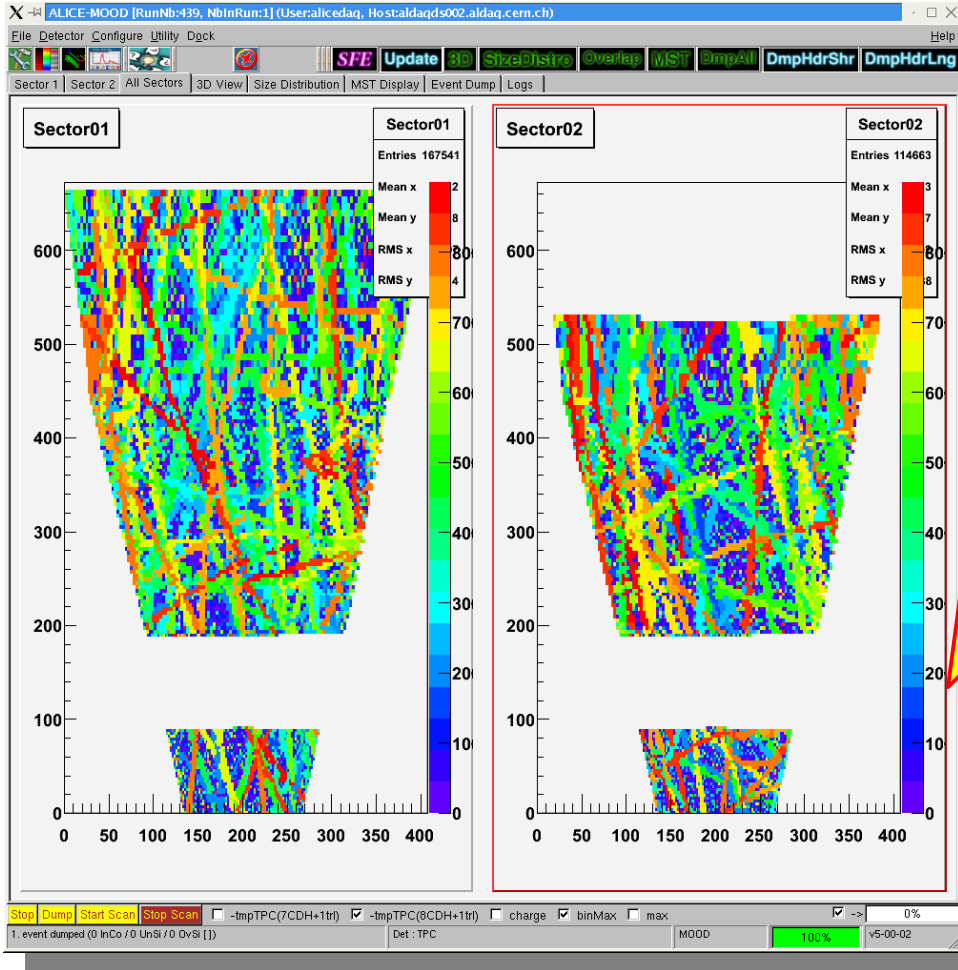


# Programlama mı Tasarım mı ?

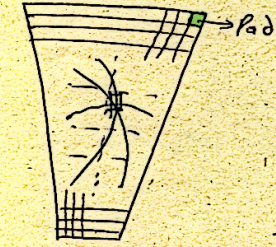
Düşün, tasarla, yaz, uygula ve sonra programlamaya başla



2002'den bugüne



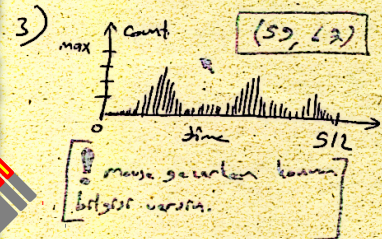
inner side of TPC



Dilim Selected

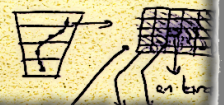
Dilim kliklelendiğinde 2. no.lu pencere oluşacaktır. Save (PS), Exit, Help, Print butonları oluşur. İz göstergesi yoktur. Cluster göstergesi yoktur. Mouse üzerine geldiğinde parlaklık ve kısa bilgi vermesi istiyorum.

(mouse-1) Pad kliklelendiğinde 3. no.lu pencere oluşur. Save (PS), Exit, Print, Help butonları burada oluşur. Fined padlar burada gösterilmeli ve charge collection göre renk bilgisine ile (blurred olacak tabiri) iz göstermek hale getirilmelidir. Update butonu ile gösterisi güncellenmelidir. Mouse üzerine gelen pa parlakmalıdır (mouse-3) sağ kliklelendiğinde fiş kovanı bilgisini gösterilmelidir.



Charge collection bilgisini zamanın fonksiyonu olarak gösterilmelidir. Update butonu başında aynı pad için yazılan dosya tabiri alınmalı ve spectrum güncellenmelidir. Save, Print, Exit, Help butonları zaten oluşur. Hangi fiziksel pad olduğu ve nun ne gibi şeyler de basılmalıdır.

4) DATE 'den gelen cluster bilgisini pad'a orta klikleyle alınmalıdır. Data içindeki cluster algoritması ve ayarlarıyla kucaklanabilir ve daha net fikir için



# Aksam Sefası

Ertesi günün sabahına hazırlanması beklenen akşamlik ödevler

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



*Dersimiz bitti !..*

\*

# ROOT - Kısa Bir Tanıtım

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

## Hedeflerimiz...

- 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

\*