
CERN

Making the TTreeReader interface more accessible

Akos Hajdu

Summer student, 08/06/2015 – 14/08/2015

Supervisors: Bertrand Bellenot, Axel Naumann

Introduction

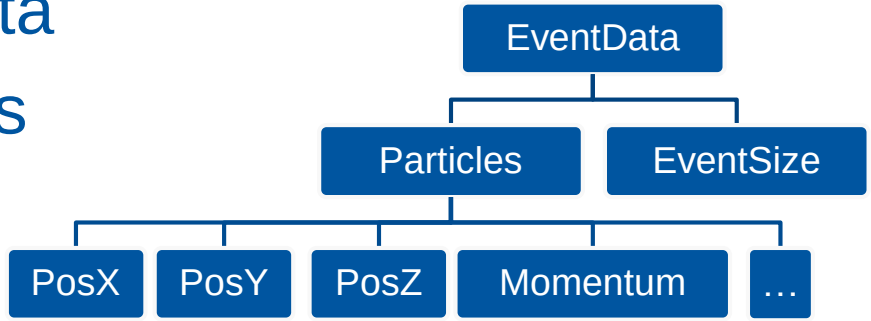
- Working on ROOT
  - Main toolkit in High Energy Physics
  - Developed at CERN PH-SFT

Introduction

- Working on ROOT
  - Main toolkit in High Energy Physics
  - Developed at CERN PH-SFT
- Purpose of the project
  - TTreeReader interface introduced in ROOT 6
  - Make it more accessible to users

What is a TTree?

- Data structure of ROOT
  - Large quantities of data
  - Organized in branches
  - Efficient disk usage
  - Optimized I/O



Accessing a TTree

- Before ROOT 6
  - Variables
  - Branch addresses

```
Double_t  fParticles_fPosX[100];
Int_t     fEventSize;

TBranch  *b_fParticles_fPosX;
TBranch  *b_fEventSize;

fChain->SetBranchAddress("fParticles.fPosX",
          fParticles_fPosX, &b_fParticles_fPosX);
fChain->SetBranchAddress("fEventSize",
          &fEventSize, &b_fEventSize);
```

Accessing a TTree

- Before ROOT 6
  - Variables
  - Branch addresses
- After ROOT 6
  - TTreeReader
    - Values, arrays
    - Simple, type safe

```
Double_t  fParticles_fPosX[100];
Int_t     fEventSize;

TBranch  *b_fParticles_fPosX;
TBranch  *b_fEventSize;

fChain->SetBranchAddress("fParticles.fPosX",
          fParticles_fPosX, &b_fParticles_fPosX);
fChain->SetBranchAddress("fEventSize",
          &fEventSize, &b_fEventSize);
```

```
TTreeReaderArray<Double_t>
   fParticles_fPosX(fReader, "fParticles.fPosX");
TTreeReaderValue<Int_t>
   fEventSize(fReader, "fEventSize");
```

Accessing a TTree

- Before ROOT 6
  - Variables
  - Branch addresses
- After ROOT 6
  - TTreeReader
    - Values, arrays
    - Simple, type safe

```
Double_t  fParticles_fPosX[100];
Int_t     fEventSize;

TBranch  *b_fParticles_fPosX;
TBranch  *b_fEventSize;

fChain->SetBranchAddress("fParticles.fPosX",
          fParticles_fPosX, &b_fParticles_fPosX);
fChain->SetBranchAddress("fEventSize",
          &fEventSize, &b_fEventSize);
```

void*

```
TTreeReaderArray<Double_t>
   fParticles_fPosX(fReader, "fParticles.fPosX");
TTreeReaderValue<Int_t>
   fEventSize(fReader, "fEventSize");
```

Accessing a TTree

- Before ROOT 6
  - Variables
  - Branch addresses
- After ROOT 6
  - TTreeReader
    - Values, arrays
    - Simple, type safe

```
Double_t  fParticles_fPosX[100];
Int_t     fEventSize;

TBranch  *b_fParticles_fPosX;
TBranch  *b_fEventSize;

fChain->SetBranchAddress("fParticles.fPosX",
          fParticles_fPosX, &b_fParticles_fPosX);
fChain->SetBranchAddress("fEventSize",
          &fEventSize, &b_fEventSize);
```

void*

Type safe

```
TTreeReaderArray<Double_t>
   fParticles_fPosX(fReader, "fParticles.fPosX");
TTreeReaderValue<Int_t>
   fEventSize(fReader, "fEventSize");
```

Accessing a TTree

- Common task: loop over elements

Accessing a TTree

- Common task: loop over elements
  - Initialize
  - Process each element (possibly parallel)
  - Terminate, finalize

Accessing a TTree

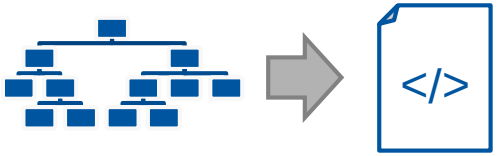
- Common task: loop over elements
  - Initialize
  - Process each element (possibly parallel)
  - Terminate, finalize
- Repetative pattern
  - Skeletons can be generated

Accessing a TTree

- Analysis skeletons (TSelector)

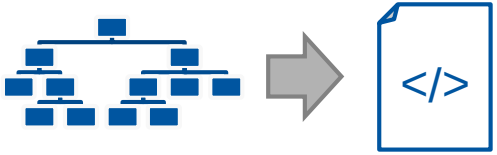
Accessing a TTree

- Analysis skeletons (TSelector)
  - Can be generated from TTrees



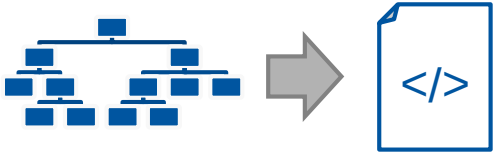
Accessing a TTree

- Analysis skeletons (TSelector)
  - Can be generated from TTrees
  - Variable declarations for accessing branches
  - Empty functions to be filled by user



Accessing a TTree

- Analysis skeletons (TSelector)
  - Can be generated from TTrees
  - Variable declarations for accessing branches
  - Empty functions to be filled by user



Accessing a TTree

- Existing skeleton generators
  - Some issues (collections, complex trees)
  - „Before ROOT 6” way of accessing the tree

Accessing a TTree

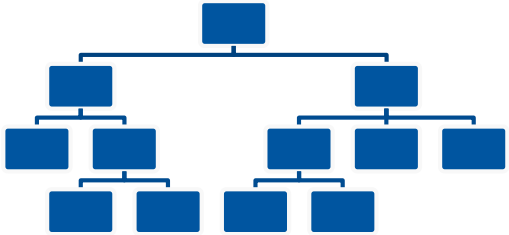
- Existing skeleton generators
  - Some issues (collections, complex trees)
  - „Before ROOT 6” way of accessing the tree
- Aim of my project
  - Generate skeletons that use TTreeReader
  - TTreeReader will be more accessible

My main tasks

1. Write a code generator

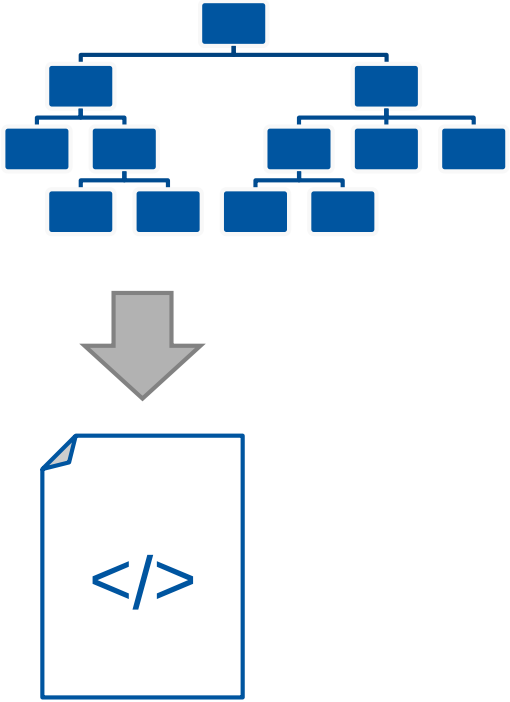
My main tasks

1. Write a code generator



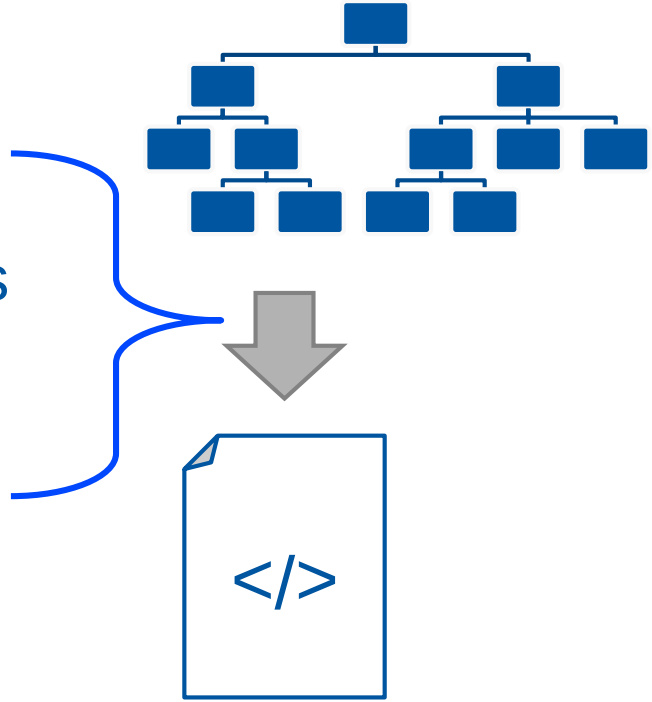
My main tasks

1. Write a code generator



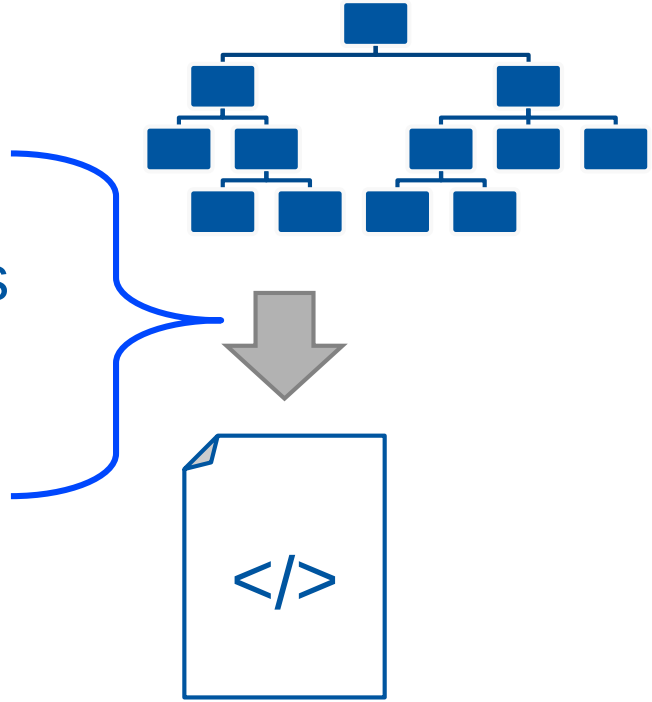
My main tasks

1. Write a code generator
   - Traverse and analyze tree
     - Extract (sub-)branches, leaves



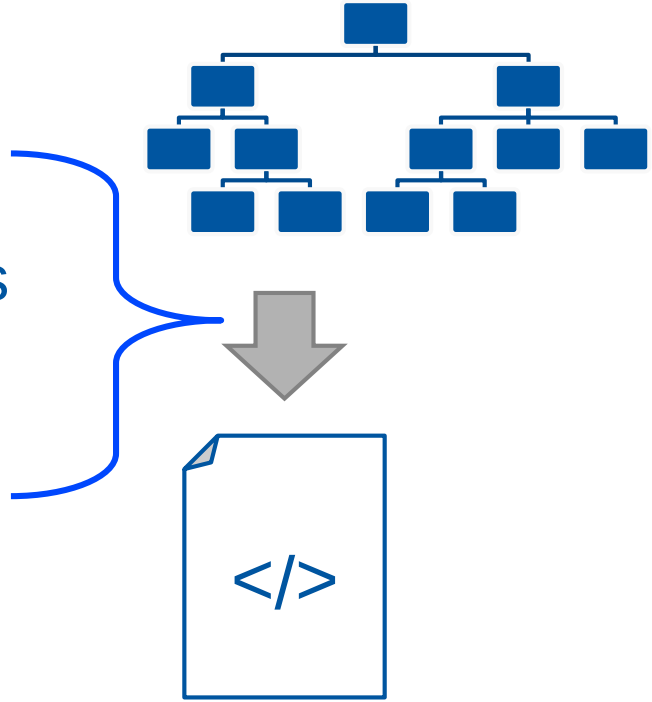
My main tasks

1. Write a code generator
   - Traverse and analyze tree
     - Extract (sub-)branches, leaves
       - Name, type, collection, etc…



My main tasks

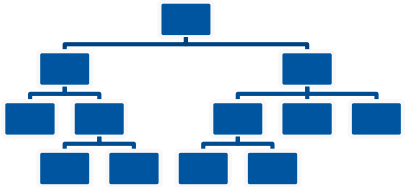
1. Write a code generator
   - Traverse and analyze tree
     - Extract (sub-)branches, leaves
       - Name, type, collection, etc…
   - Generate output code



My main tasks

2. Testing

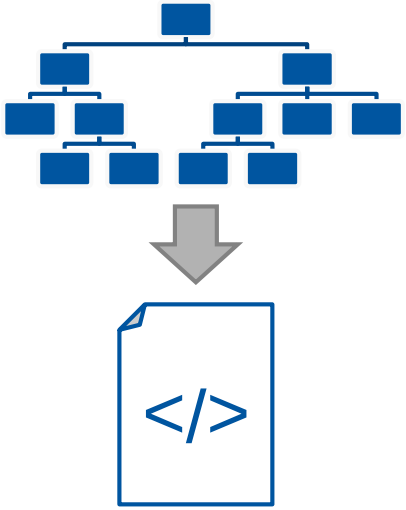
My main tasks



2. Testing
    - Creating trees
        - Simple to complex
        - Collections, nested classes, …

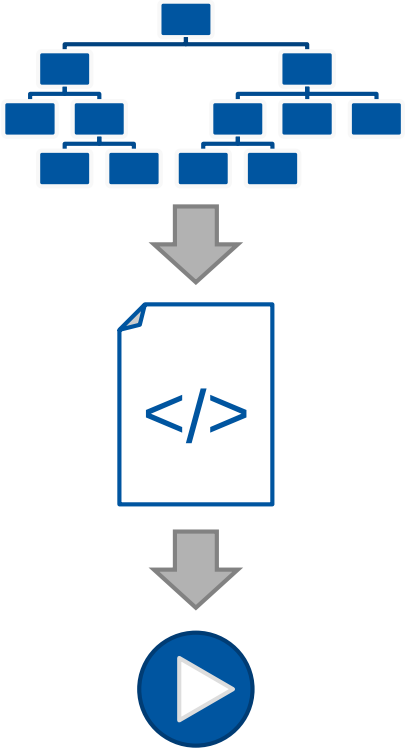
My main tasks

2. Testing
   - Creating trees
     - Simple to complex
     - Collections, nested classes, …
   - Generating skeletons



My main tasks

2. Testing
   - Creating trees
     - Simple to complex
     - Collections, nested classes, …
   - Generating skeletons
   - Accessing the tree
     - Filling the skeleton, running the code



Challenges

- Analyzing the tree
  - Existing code generators
    - MakeSelector: many cases not handled
    - MakeProxy: complex code (~4000 LOC)

Challenges

- Analyzing the tree
  - Existing code generators
    - MakeSelector: many cases not handled
    - MakeProxy: complex code (~4000 LOC)
  - Many special cases
    - Leaflists, nested classes, various collections, nested collections, …

Challenges

- Accessing the tree
  - Issues with TTreeReaderArray
    - Found: ~7, fixed: ~3, working on the remaining

Challenges

- Accessing the tree
  - Issues with TTreeReaderArray
    - Found: ~7, fixed: ~3, working on the remaining
    - Examples:
      - Accessing a vector of primitive type
      - Vector of a class containing an array

MakeSelector example

```
class Selector : public TSelector {
  TTree          *fChain;

  const Int_t kMaxfParticles = 10;

  Int_t     fParticles_;
  Double_t  fParticles_fPosX[kMaxfParticles];
  Double_t  fParticles_fPosY[kMaxfParticles];
  Double_t  fParticles_fPosZ[kMaxfParticles];
  Int_t     fEventSize;

  TBranch  *b_Event_branch_fParticles_;
  TBranch  *b_fParticles_fPosX;
  TBranch  *b_fParticles_fPosY;
  TBranch  *b_fParticles_fPosZ;
  TBranch  *b_Event_branch_fEventSize;
  ...
}
```

```
class Selector : public TSelector {
  TTree          *fChain;

  const Int_t kMaxfParticles = 10;

  Int_t     fParticles_;
  Double_t  fParticles_fPosX[kMaxfParticles];
  Double_t  fParticles_fPosY[kMaxfParticles];
  Double_t  fParticles_fPosZ[kMaxfParticles];
  Int_t     fEventSize;

  TBranch  *b_Event_branch_fParticles_;
  TBranch  *b_fParticles_fPosX;
  TBranch  *b_fParticles_fPosY;
  TBranch  *b_fParticles_fPosZ;
  TBranch  *b_Event_branch_fEventSize;
  ...
}
```

Array size

MakeSelector example

MakeSelector example

```
class Selector : public TSelector {
  TTree          *fChain;

  const Int_t kMaxfParticles = 10;

  Int_t     fParticles_;
  Double_t  fParticles_fPosX[kMaxfParticles];
  Double_t  fParticles_fPosY[kMaxfParticles];
  Double_t  fParticles_fPosZ[kMaxfParticles];
  Int_t     fEventSize;

  TBranch  *b_Event_branch_fParticles_;
  TBranch  *b_fParticles_fPosX;
  TBranch  *b_fParticles_fPosY;
  TBranch  *b_fParticles_fPosZ;
  TBranch  *b_Event_branch_fEventSize;
  ...
}
```

Array size

Leaves only

```
class Selector : public TSelector {
   TTree          *fChain;

   const Int_t kMaxfParticles = 10;

   Int_t     fParticles_;
   Double_t  fParticles_fPosX[kMaxfParticles];
   Double_t  fParticles_fPosY[kMaxfParticles];
   Double_t  fParticles_fPosZ[kMaxfParticles];
   Int_t     fEventSize;

   TBranch  *b_Event_branch_fParticles_;
   TBranch  *b_fParticles_fPosX;
   TBranch  *b_fParticles_fPosY;
   TBranch  *b_fParticles_fPosZ;
   TBranch  *b_Event_branch_fEventSize;
   ...
}
```

Array size

Leaves only

MakeSelector example

```
void Selector::Init(TTree *tree) {
   if (!tree) return;
   fChain = tree;
   fChain->SetMakeClass(1);

   fChain->SetBranchAddress("fParticles", &fParticles_,
&b_Event_branch_fParticles_);
   fChain->SetBranchAddress("fParticles.fPosX",
fParticles_fPosX, &b_fParticles_fPosX);
   fChain->SetBranchAddress("fParticles.fPosY",
fParticles_fPosY, &b_fParticles_fPosY);
   fChain->SetBranchAddress("fParticles.fPosZ",
fParticles_fPosZ, &b_fParticles_fPosZ);
   fChain->SetBranchAddress("fEventSize", &fEventSize,
&b_Event_branch_fEventSize);
}
```

Initialization

```
class Selector : public TSelector {
  TTree          *fChain;

  const Int_t kMaxfParticles = 10;

  Int_t     fParticles_;
  Double_t  fParticles_fPosX[kMaxfParticles];
  Double_t  fParticles_fPosY[kMaxfParticles];
  Double_t  fParticles_fPosZ[kMaxfParticles];
  Int_t     fEventSize;

  TBranch  *b_Event_branch_fParticles_;
  TBranch  *b_fParticles_fPosX;
  TBranch  *b_fParticles_fPosY;
  TBranch  *b_fParticles_fPosZ;
  TBranch  *b_Event_branch_fEventSize;
  ...
}
```

Array size

Leaves only

```
void Selector::Init(TTree *tree) {
   if (!tree) return;
   fChain = tree;
   fChain->SetMakeClass(1);

   fChain->SetBranchAddress("fParticles", &fParticles_,
&b_Event_branch_fParticles_);
   fChain->SetBranchAddress("fParticles.fPosX",
fParticles_fPosX, &b_fParticles_fPosX);
   fChain->SetBranchAddress("fParticles.fPosY",
fParticles_fPosY, &b_fParticles_fPosY);
   fChain->SetBranchAddress("fParticles.fPosZ",
fParticles_fPosZ, &b_fParticles_fPosZ);
   fChain->SetBranchAddress("fEventSize", &fEventSize,
&b_Event_branch_fEventSize);
}
```

MakeSelector example

Initialization

```
Bool_t Selector::Process(Long64_t entry) {
   b_fParticles_fPosX->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosX[i] << endl;
   b_fParticles_fPosY->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosY[i] << endl;
   b_fParticles_fPosZ->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosZ[i] << endl;
}
```

Accessing the data

```
class Selector : public TSelector {
  TTree          *fChain;

  const Int_t kMaxfParticles = 10;

  Int_t     fParticles_;
  Double_t  fParticles_fPosX[kMaxfParticles];
  Double_t  fParticles_fPosY[kMaxfParticles];
  Double_t  fParticles_fPosZ[kMaxfParticles];
  Int_t     fEventSize;

  TBranch  *b_Event_branch_fParticles_;
  TBranch  *b_fParticles_fPosX;
  TBranch  *b_fParticles_fPosY;
  TBranch  *b_fParticles_fPosZ;
  TBranch  *b_Event_branch_fEventSize;
  ...
}
```

Array size

Leaves only

```
void Selector::Init(TTree *tree) {
   if (!tree) return;
   fChain = tree;
   fChain->SetMakeClass(1);

   fChain->SetBranchAddress("fParticles", &fParticles_,
&b_Event_branch_fParticles_);
   fChain->SetBranchAddress("fParticles.fPosX",
fParticles_fPosX, &b_fParticles_fPosX);
   fChain->SetBranchAddress("fParticles.fPosY",
fParticles_fPosY, &b_fParticles_fPosY);
   fChain->SetBranchAddress("fParticles.fPosZ",
fParticles_fPosZ, &b_fParticles_fPosZ);
   fChain->SetBranchAddress("fEventSize", &fEventSize,
&b_Event_branch_fEventSize);
}
```

MakeSelector example

Initialization

```
Bool_t Selector::Process(Long64_t entry) {
   b_fParticles_fPosX->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosX[i] << endl;
   b_fParticles_fPosY->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosY[i] << endl;
   b_fParticles_fPosZ->GetEntry(entry);
   for (Int_t i = 0; i < fParticles_; ++i) cout << fParticles_fPosZ[i] << endl;
}
```

Load explicitly

Accessing the data

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader     fReader;

   TTreeReaderArray<Double_t> fParticles_fPosX;
   TTreeReaderArray<Double_t> fParticles_fPosY;
   TTreeReaderArray<Double_t> fParticles_fPosZ;
   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),
            fParticles_fPosX(fReader, "fParticles.fPosX"),
            fParticles_fPosY(fReader, "fParticles.fPosY"),
            fParticles_fPosZ(fReader, "fParticles.fPosZ"),
            fEventSize(fReader, "fEventSize") { }
   ...
};
```

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader     fReader;

   TTreeReaderArray<Double_t> fParticles_fPosX;
   TTreeReaderArray<Double_t> fParticles_fPosY;
   TTreeReaderArray<Double_t> fParticles_fPosZ;
   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),
            fParticles_fPosX(fReader, "fParticles.fPosX"),
            fParticles_fPosY(fReader, "fParticles.fPosY"),
            fParticles_fPosZ(fReader, "fParticles.fPosZ"),
            fEventSize(fReader, "fEventSize") { }
   ...
};
```

Type safe

Leaves

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader     fReader;

   TTreeReaderArray<Double_t> fParticles_fPosX;
   TTreeReaderArray<Double_t> fParticles_fPosY;
   TTreeReaderArray<Double_t> fParticles_fPosZ;
   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),
            fParticles_fPosX(fReader, "fParticles.fPosX"),
            fParticles_fPosY(fReader, "fParticles.fPosY"),
            fParticles_fPosZ(fReader, "fParticles.fPosZ"),
            fEventSize(fReader, "fEventSize") { }
   ...
};
```

Type safe

Leaves

```
void Selector::Init(TTree *tree)
{
   fReader.SetTree(tree);
}
```

Initialization

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader     fReader;

   TTreeReaderArray<Double_t> fParticles_fPosX;
   TTreeReaderArray<Double_t> fParticles_fPosY;
   TTreeReaderArray<Double_t> fParticles_fPosZ;
   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),
            fParticles_fPosX(fReader, "fParticles.fPosX"),
            fParticles_fPosY(fReader, "fParticles.fPosY"),
            fParticles_fPosZ(fReader, "fParticles.fPosZ"),
            fEventSize(fReader, "fEventSize") { }
   ...
};
```

Type safe

Leaves

```
void Selector::Init(TTree *tree)
{
   fReader.SetTree(tree);
}
```

Initialization

Accessing the data

```
Bool_t Selector::Process(Long64_t entry)
{
   fReader.SetEntry(entry);
   for(Int_t i = 0; i < fParticles_fPosX.GetSize(); ++i) cout << fParticles_fPosX[i] << endl;
   for(Int_t i = 0; i < fParticles_fPosY.GetSize(); ++i) cout << fParticles_fPosY[i] << endl;
   for(Int_t i = 0; i < fParticles_fPosZ.GetSize(); ++i) cout << fParticles_fPosZ[i] << endl;
}
```

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader     fReader;

   TTreeReaderArray<Double_t> fParticles_fPosX;
   TTreeReaderArray<Double_t> fParticles_fPosY;
   TTreeReaderArray<Double_t> fParticles_fPosZ;
   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),
            fParticles_fPosX(fReader, "fParticles.fPosX"),
            fParticles_fPosY(fReader, "fParticles.fPosY"),
            fParticles_fPosZ(fReader, "fParticles.fPosZ"),
            fEventSize(fReader, "fEventSize") { }
   ...
};
```

Type safe

Leaves

```
void Selector::Init(TTree *tree)
{
   fReader.SetTree(tree);
}
```

Initialization

```
Bool_t Selector::Process(Long64_t entry)
{
   fReader.SetEntry(entry);
   for(Int_t i = 0; i < fParticles_fPosX.GetSize(); ++i) cout << fParticles_fPosX[i] << endl;
   for(Int_t i = 0; i < fParticles_fPosY.GetSize(); ++i) cout << fParticles_fPosY[i] << endl;
   for(Int_t i = 0; i < fParticles_fPosZ.GetSize(); ++i) cout << fParticles_fPosZ[i] << endl;
}
```

Set entry, read on demand

Accessing the data

TTreeReader example

```
class Selector : public TSelector {

   TTreeReader      fReader;

   TTreeReaderArray<Particle> fParticles;

   TTreeReaderValue<Int_t> fEventSize;

   Selector(TTree * /*tree*/ =0) : fChain(0),

            fParticles(fReader, "fParticles"),

            fEventSize(fReader, "fEventSize") { }
   ...
};
```

Type safe

Access node

```
void Selector::Init(TTree *tree)
{
   fReader.SetTree(tree);
}
```

Initialization

```
Bool_t Selector::Process(Long64_t entry)
{
   fReader.SetEntry(entry);
   for(Int_t i = 0; i < fParticles.GetSize(); ++i) cout << fParticles[i].fPosX << endl;
   for(Int_t i = 0; i < fParticles.GetSize(); ++i) cout << fParticles[i].fPosY << endl;
   for(Int_t i = 0; i < fParticles.GetSize(); ++i) cout << fParticles[i].fPosZ << endl;
}
```

Set entry, read on demand

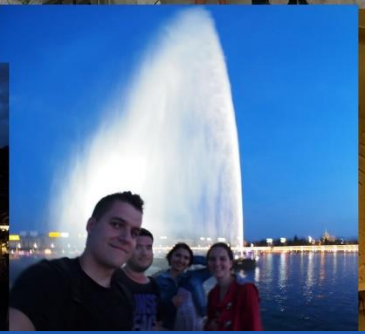
Accessing the data

Besides work







Conclusion

What is done (8 weeks)

- Created test trees
- Examined TTreeReader
- Examined the old way of generating skeletons
- Implemented TTreeReader code generator
- Found issues with TTreeReaderArray

Conclusion

What is done (8 weeks)

- Created test trees
- Examined TTreeReader
- Examined the old way of generating skeletons
- Implemented TTreeReader code generator
- Found issues with TTreeReaderArray

What are the plans (2 weeks)

- Finalize the new interface
- Solve issues with TTreeReaderArray

Conclusion

What is done (8 weeks)

- Created test trees
- Examined TTreeReader
- Examined the old way of generating skeletons
- Implemented TTreeReader code generator
- Found issues with TTreeReaderArray

What are the plans (2 weeks)

- Finalize the new interface
- Solve issues with TTreeReaderArray

root.cern.ch
github.com/hajduakos
akos.hajdu@cern.ch