Plans for RootIO.jl

Google Summer of Code, 2024

Yash Solanki[1] Philippe Gras[2] Pere Mato Vila[3]

[1]Indian Institute of Technology, Delhi
[2]Université Paris-Saclay
[3]CERN, EP-SFT

Julia HEP WG meeting
20 June, 2024

Introduction

Currently, if you want to create a TTree in a root file, you have to write everything manually, which is not always convenient for the users

```
1
2       using ROOT
3       println("Creating a ROOT file with a TTree filled with scalars.\n")
4       nevts = 10
5       f = ROOT.TFile!Open("test1.root", "RECREATE")
6       t = ROOT.TTree("tree", "tree")
7       a = fill(0)
8       Branch(t, "a", a, 32000, 99)
9       for i in 1:nevts
10          a[] = i
11          println("Writing value ", a[])
12          Fill(t)
13      end
14      Write(t)
15      Close(f)
16
```

Inputs using UnROOT.jl

Reading back from the TTree is supported by UnROOT, which simplifies taking input from TFiles. It is written entirely in Julia without any dependence on C++ or Python.

```
1
2       using UnROOT
3
4       println("Reading back the file created with LazyTree of UnROOT.jl")
5
6       t = LazyTree("test2.root", "tree")
7       display(t)
8
```

What about Output?

Through my Google Summer of Code project, I will create the RootIO.jl module, which would simplify writing to the root files.

```
1
2       import ROOTIO, DataFrames
3       nrows = 10
4       df = DataFrame(x = rand(nrows), y = rand(nrows), z = rand(nrows), v = [rand(5)
    for irow in 1:nrows])
5       ROOTIO.Write("data.root", "mytree", "mytreetitle", df)
6
```

RootIO.jl will be created in 4 steps, with a deliverable at each step:

- Baseline support
- Validation using UnROOT to read back values
- Implementation of extra features
- Unit tests and documentation

A more detailed overview of the module can be found in the Julia-TFile document.

Extra features of RootIO.jl

The RootIO.jl module will support the writing of:

- Plain Vector fields of Julia in the data structs: stored as std::vector or C-array or both
- Nested structs: Fields of a struct, which are themselves structs will be supported.
- The Julia types Tuple (and NTuple)
- Vector of structs and Vector of Vectors
- Storing instances of classes that does not inherit from TObject

Timeline

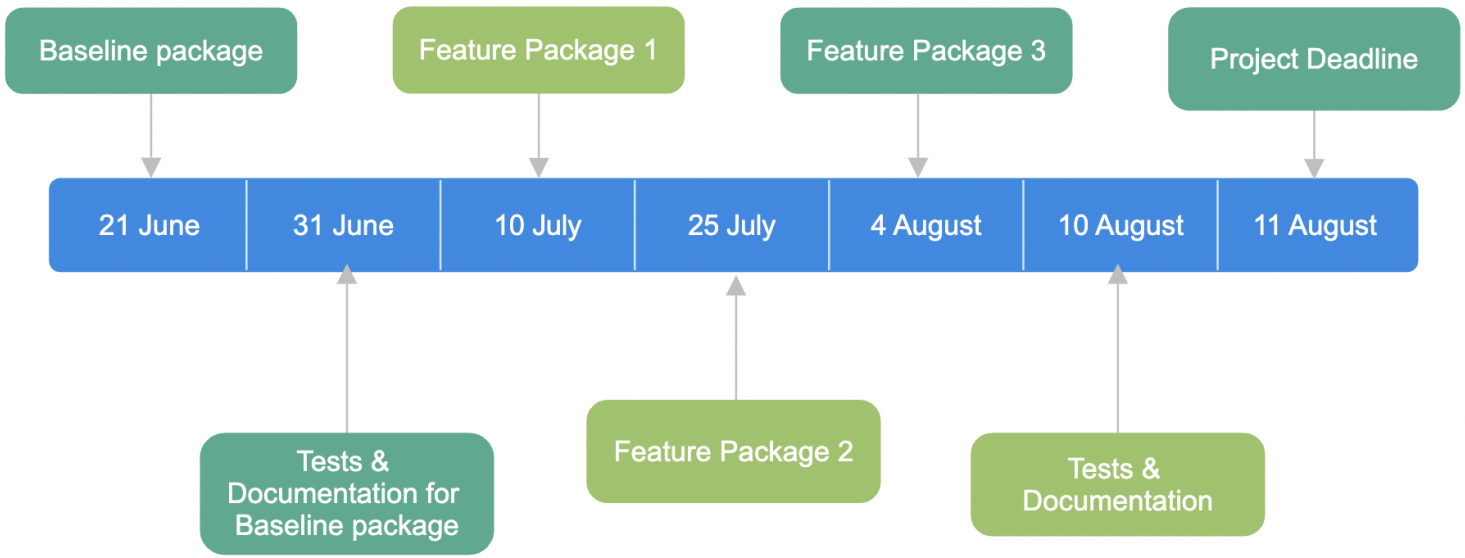

Figure: Timeline of GSoC project

Current status of RootIO.jl

The Pull Request baseline support for writing to TTree has been opened. The following types will be supported when the changes will be merged:

Julia type	C++ type[a]	TTree type code	Description
`String`	`char*`	`C`	a character string.
`Int8`	`Char_t`	`B`	an 8 bit signed integer.
`UInt8`	`UChar_t`	`b`	an 8 bit unsigned integer
`Int16`	`Short_t`	`S`	a 16 bit signed integer
`UInt16`	`UShort_t`	`s`	a 16 bit unsigned integer
`Int32`	`Int_t`	`I`	a 32 bit signed integer
`UInt32`	`UInt_t`	`i`	a 32 bit unsigned integer
`Float32`	`Float_t`	`F`	a 32 bit floating point
`Half32`[b]	`Float16_t`	`f`	32 bits in memory, 16 bits on disk
`Float64`	`Double_t`	`D`	a 64 bit floating point
`Double32`[c]	`Double32_t`	`d`	64 bits in memory, 32 on disk
`Int64`	`Long64_t`	`L`	a 64 bit signed integer
`UInt64`	`ULong64_t`	`l`	a 64 bit unsigned integer
`Int64`	`Long_t`	`G`	a long signed integer, stored as 64 bit
`UInt64`	`ULong_t`	`g`	a long unsigned integer, stored as 64 bit
`Bool`	`bool`	`O`[d]	a boolean
`StdVector{T}`	`std::vector{T}`	N/A	Vector of elements of any of the above type.

Figure: Supported types in the baseline package

Examples

The following example demonstrates how we can create a custom struct and write it to the TTree

```
1
2       import RootIO, ROOT
3       using DataFrames
4
5       mutable struct Event
6           x::Float64
7           y::Float64
8           z::Float64
9           v::Vector{Float64}
10      end
11
12      f = ROOT.TFile!Open("data.root", "RECREATE")
13      Event() = Event(0., 0., 0., Float64[])
14      tree = RootIO.TTree(f, "mytree", "mytreetitle", Event)
15      e = Event()
16      for i in 1:10
17          e.x, e.y, e.z = rand(3)
18          v = rand(5)
19          RootIO.Fill(tree, e)
20      end
21      RootIO.Write(tree)
22      ROOT.Close(f)
23
```

Conclusion

- A new module called RootIO.jl will be created, which will streamline the writing of data to TFiles
- Support for the writing of all primitive types, structs, julia vectors, tuples, nested structs and custom objects will be provided
- Future goal- Add support for the experimental RNtuple that will replace the TTree
- Thanks to Pere and Philippe for help writing the module and mentoring me for Google Summer of Code!