Structuring the output of ROOT file and file merging

March 18, 2021

In the fourth ROOT exercise we cover the following frequently used classes:

1. TList (`https://root.cern.ch/doc/master/classTList.html`)
2. TDirectoryFile (`https://root.cern.ch/doc/v612/classTDirectoryFile.html`)
3. TFileMerger (`https://root.cern.ch/doc/master/classTFileMerger.html`)

The main point behind this exercise is to learn about typical structuring of your output ROOT file, which is for instance adopted in ALICE Collaboration. Besides, we also cover the essence of 'modus operandi' which is nowadays encountered regularly in high energy physics (e.g. running in distributed mode with 1000+ CPUs over some massive datasets, collecting and merging of all output files, etc.).

Usually, you develop a bunch of classes in C++ or ROOT for the data analysis (or analyses tasks, if the official analysis framework of ALICE within AliROOT is used). Each class produces some output (histograms, profiles, etc.), and within each class, you might have different member functions producing separate histograms, profiles, etc. How then you would structure internally all this mess in your final physical ROOT file? This is my suggestion, which is also supported within the ALICE analysis framework:

1. output of each class store in a TDirectoryFile
2. output of each class' member function store in a dedicated TList

Therefore, the content of output ROOT file ideally would have internal structure demonstrated with the following code snippet:

```
{
 // Physical ROOT file:
 TFile *file = new TFile("test.root","recreate");

 // Output of first class will be saved here:
 TDirectoryFile *dirFile_1 = new TDirectoryFile("dirFile_1","dirFile_1_title");
 // Output of first member function in the first class will be saved here:
 TList *list_11 = new TList();
 list_11->SetName("list_11_name");
 // Output of second member function in the first class will be saved here:
 TList *list_12 = new TList();
 list_12->SetName("list_12_name");
 // ...

 // Output of second class will be saved here:
 TDirectoryFile *dirFile_2 = new TDirectoryFile("dirFile_2","dirFile_2_title");
 // Output of first member function in the second class will be saved here:
 TList *list_21 = new TList();
 list_21->SetName("list_21_name");
 // ...

 // Some example output objects:
 TH1D *hist_111 = new TH1D("hist_111","hist_111_title",10,0.,1.);
 hist_111->SetDirectory(0); // by default, upon creation, each histogram is automatically saved in the currently opened file. You disable that with this line.
 list_11->Add(hist_111); // add histogram to its TList
 TH1D *hist_112 = new TH1D("hist_112","hist_112_title",100,0.,100.);
 hist_112->SetDirectory(0); // by default, upon creation, each histogram is automatically saved in the currently opened file. You disable that with this line.
 list_11->Add(hist_112); // add histogram to its TList
 // Finally, add TList to its TDirectoryFile:
 dirFile_1->Add(list_11);

 // Some more example output objects:
 TProfile *pro_211 = new TProfile("pro_211","pro_211_title",10,-5.,5.);
 pro_211->SetDirectory(0); // by default, upon creation, each profile is automatically saved in the currently opened file. You disable that with this line.
 list_21->Add(pro_211); // add profile to its TList
 // Finally, add TList to its TDirectoryFile:
 dirFile_2->Add(list_21);

 // Finally, dump each TDirectoryFile in the physical ROOT file:
 dirFile_1->Write(dirFile_1->GetName(),TObject::kSingleKey);
 dirFile_2->Write(dirFile_2->GetName(),TObject::kSingleKey);

}
```

Question 1: Please execute this code snippet, and with the TBrowser inspect the internal structure of the resulting ROOT file "test.root". Study this code snippet, because this is a fairly generic example, and see what you can modify, and what you cannot, without encountering trouble...

Question 2: Start with a blank file and develop a code that will make a histogram with 10 bins and fill it with 10000 points sampled from your favorite function. Take care that histogram is saved in the TList, which is saved in the TDirectoryFile, which is saved in the physical file (as in Q1) above).

Question 3: Make 5 subdirectories, SUBDIR_0, SUBDIR_1, ..., SUBDIR_4, copy your code developed in 2) in each subdirectory, and execute. You should get 5 output ROOT files with the same internal structure and the same name in 5 subdirectories. How to prevent sampling in each case to be exactly the same (i.e. you want 5 independent toy Monte Carlo simulations in this example, and not 5 identical copies of the same one)?

Question 4: The Q3 is the simplest example you can think of, but the point I would like to illustrate with it is clear: In reality, you will utilize a huge number of computers to analyze the large LHC datasets, or to run in parallel locally Monte Carlo simulations, and each process will produce the output ROOT file with the same internal structure and the same name, sitting in some subdirectory. Now you want to merge all those low statistics files to get the final, large statistics merged output file, after all jobs have finished. How you would merge output files in Q3 sitting in SUBDIR_0, SUBDIR_1, ..., SUBDIR_4 into one, large statistics, ROOT file? Please develop that code. Hint: The class TFileMerger might be very helpful...

Question 5: Validation of merging: Please develop a code snippet which will access the content of let's say first bin of each histogram sitting in the files in SUBDIR_0, SUBDIR_1, ..., SUBDIR_4, and sum up these contents programmatically. Is the result the same as the content of the first bin in the merged, large statistics file?
Hint: Given your internal file structure, to fetch the pointer to histogram from an external file, you may find the following code snippet useful:

```
TFile *outputFile = TFile::Open(<externalPathToYourFile>,"READ");
// outputFile->ls(); // prints the name of objects stored in this file
TDirectoryFile *directoryFile = dynamic_cast<TDirectoryFile*>(outputFile->Get(<nameOfYourTDirectoryFile>));
TList *list = dynamic_cast<TList*>(directoryFile->Get(<nameOfYourList>));
TH1D *hist = dynamic_cast<TH1D*>(list->FindObject("<nameOfYourHistograms>"));
```

Alternatively, in the case the histogram was saved directly into the TFile, you can get its pointer schematically with:

```
TFile *outputFile = TFile::Open(<externalPathToYourFile>,"READ");
TH1D *hist = dynamic_cast<TH1D*>(outputFile->Get("<nameOfYourHistograms>"));
```

Please send me the code snippets you have developed for the merging in Q4 and for the validation in Q5.