RootTuple: A library enabling ROOT n-tuple output from FORTRAN HEP programs

David C. Hall
The University of Oxford

http://roottuple.hepforge.org

Version 1.0.0

Contents

1 Introduction

There are two ways in which you might use the RootTuple library. The first is as an end user, in which case you can simply follow the instructions in §2. The second is as a developer who wants to use the FORTRAN subroutines that it supplies. In this case, I give additional information about the library in the other sections.

RootTuple consists of a C++ class (which handles the ROOT output) and a Fortran/C++ interface. The interface contains a wrapper which allows its C++ functions to be used as FORTRAN subroutines (see §4), and a global pointer to an instance of the C++ class.

2 Installation

Interfacing FORTRAN to C/C++ is filled with caveats, and for this reason we use the CMake build system[1] to optimise the portability of the code. This version of RootTuple requires CMake-2.8.6 or higher. Note that RootTuple may fail at runtime when using some of the more obscure FORTRAN compilers if they do not follow the standard convention for passing character arrays as arguments.

In order to output the n-tuples, we also require that the ROOT data analysis framework is installed[2]. We recommend using ROOT-5.26 or higher. The ROOT setup script must be sourced before the installation of RootTuple.

1. Download source tarball and extract:

   ```
   wget http://www.hepforge.org/archive/roottuple/RootTuple-1.0.0.tar.gz
   tar -xzvf RootTuple-1.0.0.tar.gz
   ```

2. Build in a separate binary directory:

   ```
   mkdir build
   cd build
   cmake -DCMAKE_INSTALL_PREFIX=<prefix> ../RootTuple-1.0.0
   make
   ```

3. Install to the `<prefix>` directory:

   ```
   make install
   ```

4. The `build` directory is no longer required and can be deleted if desired.

[1] CMake is available at http://www.cmake.org/
[2] ROOT is available at http://root.cern.ch/drupal/

2.1 What if I don't have CMake-2.8.6 installed?

Firstly, I would recommend installing CMake since it is more portable and allows the source, binary and installation files to be separated easily. It also creates this documentation file if you have LaTeX installed, and a `RootTuple-config` file to make linking easier.

However, if for some reason you are not able to install CMake, I have also included a `Makefile`. Before this can be used, the file `external/FC_namemangling.h` must be copied into the `src` directory and edited as necessary (I believe the example should work for the `gfortran`, `g77` and `ifort` compilers, but I am not giving a guarantee).

3 Use within FORTRAN HEP programs

3.1 Basic usage

The minimal use of the subroutines to output a ROOT n-tuple is (see §4 for subroutine descriptions):

1. Use `ROOTINIT` during initialisation
2. For each event:
   (a) Use `ROOTADDPARTICLE` on each final state particle
   (b) Use `ROOTADDEVENT`
3. Use `ROOTCLOSE` during exit

When building your program you must link to the RootTuple library, the location of which will be supplied by the end user. Since new subroutines will be called from the program, compilation will fail if the RootTuple library is not linked. For this reason, a file `dummyroot.f` containing dummy routines has been included in the RootTuple source code. This should be built in the case when the RootTuple library has not been linked to.

The linker must also be able to find the ROOT libraries themselves. The easiest way to do this is to source the ROOT setup script, which should set the environment variables correctly.

At runtime, the end user must add the location of the RootTuple library to their `LD_LIBRARY_PATH` (`DYLD_LIBRARY_PATH` for MacOS). The same is also true for the ROOT libraries (again it is easiest to source the ROOT setup script.)

A `RootTuple-config` file is also installed in the `bin` directory when RootTuple is installed using CMake. This can be used to make linking easier, since it includes information about both the RootTuple and ROOT installations used during its build.

3.2 Advanced usage

Some additional subroutines are provided to give greater control over the output files. For details, please see §4.

- The ROOTADDxxxxx subroutines allow the developer to add custom variables to the output file. Supported FORTRAN data types are DOUBLE PRECISION, REAL, INTEGER and LOGICAL which correspond to the double, float, int and bool C++ data types, respectively. Note that these variables must be set once per event, and before the ROOTADDEVENT subroutine is called.

4 List of FORTRAN subroutines provided

4.1 Basic subroutines

```
SUBROUTINE ROOTINIT(FILENAME)
CHARACTER*n FILENAME
```

This subroutine opens a new ROOT file with name equal to the string FILENAME. If FILENAME does not end in .root this will be appended automatically. The length of the string n is set within the FORTRAN code, and any trailing whitespace will be removed.

```
SUBROUTINE ROOTCLOSE
```

This subroutine closes the ROOT file.

```
SUBROUTINE ROOTADDPARTICLE(CODE,PX,PY,PZ,E)
INTEGER CODE
DOUBLE PRECISION PX,PY,PZ,E
```

This subroutine adds a particle to the event. CODE is an integer used to identify the particle, according to the PDG Monte Carlo particle numbering scheme. (PX,PY,PZ,E) is the four-momentum of the particle.

```
SUBROUTINE ROOTADDEVENT(WGT)
DOUBLE PRECISION WGT
```

This subroutine adds an event to the n-tuple with an event weight of WGT. This subroutine must be used after all the particles have been added to the event using ROOTADDPARTICLE.

4.2 Advanced subroutines

```
SUBROUTINE ROOTADDDOUBLE(VAL,BRANCH)
DOUBLE PRECISION VAL
CHARACTER*n BRANCH
```

This subroutine adds a double precision variable to the event file, with name BRANCH and value VAL. The length of the string n is set within the FORTRAN code, and any trailing whitespace will be removed.

```
SUBROUTINE ROOTADDFLOAT(VAL,BRANCH)
REAL VAL
CHARACTER*n BRANCH
```

This subroutine adds a single precision variable to the event file, with name `BRANCH` and value `VAL`. The length of the string `n` is set within the FORTRAN code, and any trailing whitespace will be removed.

```
SUBROUTINE ROOTADDINT(VAL,BRANCH)
INTEGER VAL
CHARACTER*n BRANCH
```

This subroutine adds an integer variable to the event file, with name `BRANCH` and value `VAL`. The length of the string `n` is set within the FORTRAN code, and any trailing whitespace will be removed.

```
SUBROUTINE ROOTADDBOOL(VAL,BRANCH)
LOGICAL VAL
CHARACTER*n BRANCH
```

This subroutine adds a boolean variable to the event file, with name `BRANCH` and value `VAL`. The length of the string `n` is set within the FORTRAN code, and any trailing whitespace will be removed.

```
SUBROUTINE ROOTWRITE
```

This subroutine will create a new key in the TTree (see ROOT documentation for more information). This is not required to write the data to disk.