Software Starters

Page Contents

Root-dependent package template

While writing simple macros or python scripts can be an effective way to quickly get started with ROOT, cases frequently arise where it would be most effective to create a library of compiled functions or even a compiled executable. Here is the setup of a template package for compiling ROOT-dependent C++ classes and applications.

First, determine which ROOT installation you will use.

On lxplus5 (i.e., slc5 operating system at CERN), set environment variables with values like these:

```
export ROOTSYS=/afs/cern.ch/sw/lcg/app/releases/ROOT/5.22.00d/i686-slc5-gcc43-dbg/root
export LD_LIBRARY_PATH=${ROOTSYS}/lib:/afs/cern.ch/sw/lcg/external/Python/2.5/i686-slc5-gcc43-opt
/lib:${LD_LIBRARY_PATH}
export PATH=${ROOTSYS}/bin:/afs/cern.ch/sw/lcg/external/Python/2.5/i686-slc5-gcc43-opt/bin:${PATH}
export PYTHONPATH=$ROOTSYS/lib
```

1. Create a directory where you will run ROOT and `cd` to it.
2. Download the `CompiledRootAnalysis` template package, e.g., `wget` http://mgwilson.web.cern.ch/mgwilson/Software/CompiledRootAnalysis_V01.tgz
3. `tar -xzvf CompiledRootAnalysis_V01.tgz`
4. Edit the top of `CompiledRootAnalysis/scripts/RenamePackage.sh` with a new package name and namespace for the code in your package; this will be denoted <package> below.
5. `CompiledRootAnalysis/scripts/RenamePackage.sh`
6. `cd <package>; gmake shlib; gmake apps; gmake setup; cd ..`
   Note 1: If you want to use MacOS, do `make dylib; make macapps` instead.
   Note 2: To see what commands are being executed during the compilation, delete the @ symbols preceding the commands in the GNUmakefile.
7. Execute the `bin/example` application to see that it runs correctly.
8. Start a root session to see that the shared library is loaded correctly. All of the classes and methods compiled in this package are available on the ROOT command line.

Athena-dependent package template

Compiling and running athena algorithms on the Grid allows one the flexibility to manipulate and analyze data in a specific manner. Here is the setup of a template package for building athena applications. This template contains the basic infrastructure and scripts for filtering events, writing an ntuple, submitting jobs to the Grid, and downloading and merging the output from the Grid.

Initial setup

First, select a suitable release by referencing the Offline Release Page, and create a work area, for example by downloading this script: `wget` http://mgwilson.web.cern.ch/mgwilson/Scripts/createAtlasWorkArea_15.6.9.8_CERN.sh; `chmod 755 createAtlasWorkArea_15.6.9.8_CERN.sh`

1. `cd` to the work area, and `source ./setup.sh`
2. Download the `AthenaUser` template package, e.g., `wget` http://mgwilson.web.cern.ch/mgwilson/Software/AthenaUser-00-00-00.tgz
3. `tar -xzvf AthenaUser-00-00-00.tgz`
4. Edit the top of `AthenaUser/scripts/renamePackage.sh` with a new package name and namespace for the code in your package; this will be denoted <package> below.
5. `AthenaUser/scripts/renamePackage.sh`
6. `cd PersonalAthenaPackage/cmt ; echo "PersonalAthenaPackage-00-00-00" > version.cmt; cmt config; gmake; cd .. /../`

Package overview

This package contains two Algorithms, one Tool, and three classes providing some utilities. The two Algorithms are for filtering events (`FilterEvent`) and for processing events that have passed the filter (`ProcessEvent`). The runtime configuration for these algorithms is in the file `share/configAOD_jobOptions.py`. The Tool and utilities are self explanatory.

The main application is `scripts/processAOD_trf.py`, which is configured in the file `share/runAOD_jobOptions.py`. Running `processAOD_trf.py` without arguments will give brief information about which arguments are expected.

How to run on a local file

If you are running on lxplus, first `cd` to a directory with a lot of space

- `cd /tmp/<username>`

Download an example AOD file from the Grid, and if running on simulated data, also download a conditions database file:

- `dq2-get -f AOD.129495._000001.pool.root.1 mc09_7TeV.105861.TTbar_PowHeg_Pythia.merge.AOD.e521_s765_s767_r1250_r1260/`
- `dq2-get -f DBRelease-10.3.1.tar.gz ddo.000001.Atlas.Ideal.DBRelease.v100301`

Then, to process all events in a simulated-data file, run a command like this:

- `processAOD_trf.py AOD.129495._000001.pool.root.1 True ntuple.root 0 DBRelease-10.3.1.tar.gz -1`

To process all events in a recorded-data file, download a file:

- `dq2-get -f data10_7TeV.00159086.physics_MuonswBeam.merge.AOD.f275_m548._lb0002-lb0021.1 data10_7TeV.00159086.physics_MuonswBeam.merge.AOD.f275_m548`

and run a command like this

- `processAOD_trf.py data10_7TeV.00159086.physics_MuonswBeam.merge.AOD.f275_m548._lb0002-lb0021.1 False ntuple.root 0 NONE -1`

How to submit jobs to the Grid

In the `scripts` directory, there are two files for submitting Grid jobs using `pathena`: `submitAODRec.sh` and `submitAODSim.sh`. The general configuration is at the top; the commands to run on specific datasets can be added at the bottom. The `N_FILES_TO_MERGE` setting at the top refers to the number of files that can be merged into a single file later on; see below. This merging can be useful because running jobs on the Grid may produce many output files; however, merging them all into a single file at the end may not be possible if the resulting file would be too large.

How to download and merge output files from the Grid

If you submitted the Grid jobs using the scripts above, then the file `download_recorded.sh` or `download_simulated.sh` should have been produced. Edit the `scripts/downloadDS.sh` to configure the download parameters, then run the `download_recorded.sh` or `download_simulated.sh` script to download and merge the output files.

General C++ questions and answers

- Q. What is `void*`?

C and C++ are strongly typed languages, meaning that blocks of memory are given a specific type (i.e., int, float, char, etc.) when allocated or used within a piece of code. Sometimes, it is desirable to write functions that can perform the same operations on different variable types. In C++, this is achievable through class inheritance. However, in C, this is done by telling the compiler not to check the type of the variable; this is done with the `void*` declaration. Specifically, `void*` is a pointer to untyped memory, and the programmer is free to cast this memory into any other type:

```
float fcn( void* var1, void* var2 ) {
float* f1 = (float*)var1;
float* f2 = (float*)var2;
/* return the product of these two floats */
return (f1)(*f2);
}
```

As you see, in doing this, the programmer assumes all responsibility for making sure that the types and operations are correct, something normally (and best) done by the compiler.