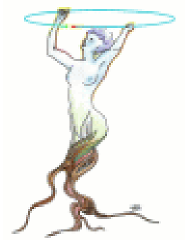
Extensions to the ROOT Plugin Manager

Fons Rademakers

Current ROOT Plugin Manager



- Plugin is simple shared library
  - No special tokens, functions, etc.
- Plugin is registered in [system].rootrc (i.e. plugin cache)

```
Plugin.TFile:   ^rfio:   TRFIOFile   RFIO   "TRFIOFile(const char*,Option_t*,const char*,Int_t)"
```

- Plugin factory via CINT call of ctor as described in rootrc (need dictionary of class).
- Class location and plugin dependencies recorded in [system].rootmap

```
Library.TMinuit:          libMinuit.so libGraf.so libHist.so libMatrix.so
```

Using a ROOT Plugin

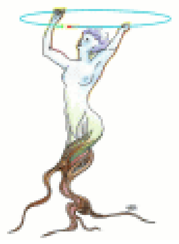
- In the code the RFIO file plugin is loaded and an TRFIOFile object is created using:

```
// name = “rfio:/cern.ch/user/r/rdm/bla.root”
TPluginHandler *h = gROOT->GetPluginManager()->FindHandler(“TFile”, name);
if (h && h->LoadPlugin() != -1)
   file = (TFile*) h->ExecPlugin(4, name, option, ftitle, compress);
```

Missing Features

- ROOT plugins are not self describing
  - The rootrc description cannot be obtained or recovered from plugin
- Manual plugin cache management
- No plugin load path override via shell variable

Make Plugins Self Describing

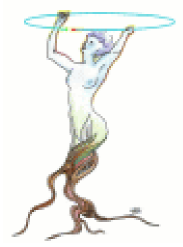


- Make plugins self describing via a simple macro to be added to the plugin source

```
ROOT_PLUGIN("1.1", "TSQLServer", "^oracle:", "TOracleServer", "Oracle", \
       "TOracleServer(const char*,const char*,const char*)", \
       "This plugin provides access to Oracle");
```

- Using “rootcint” we will automatically generate a dictionary for only the factory method
  - Scans source for “ROOT_PLUGIN” and generates dictionary for factory method

Automatic Plugin Cache Generation



- Using the new “rlibcache” utility the plugin cache will be generated (like ldconfig for Linux shared libs)
- Uses as plugin search path the “ROOT_PLUGIN_PATH” shell variable or by default the “DynamicPath” as specified in the “.rootrc” files
- The cache can be generated with absolute path names so we can run without ROOT_PLUGIN_PATH