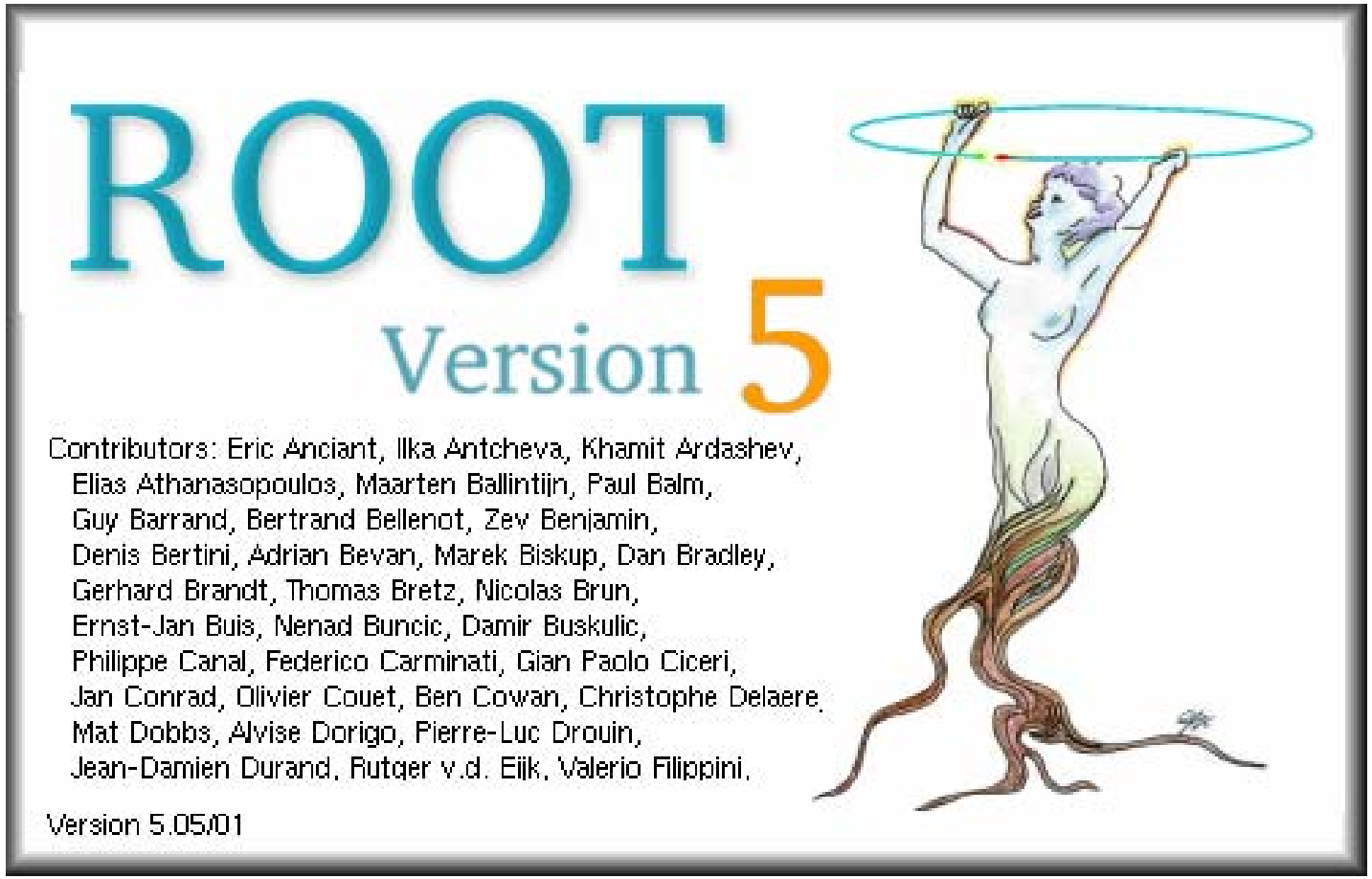
New Infrastructure Features Since ROOT 2004

Fons Rademakers

Overview

- New infrastructure features
- Miscellaneous
- Plans

New Splash Screen



TArchiveFile and TZIPFile

- TArchiveFile is an abstract class that describes an archive file containing multiple sub-files, like a ZIP or TAR archive.
- The TZIPFile class describes a ZIP archive file containing multiple ROOT sub-files. Notice that the ROOT files should not be compressed when being added to the ZIP file, since ROOT files are normally already compressed. To create the file multi.zip do:

```
zip –n root multi file1.root file2.root
```

- The ROOT files in an archive can be simply accessed like this:

```
TFile *f = TFile::Open("multi.zip#file2.root")
  or
TFile *f = TFile::Open("root://pcsalo/multi.zip#2")
```

- A TBrowser and TChain interface will follow shortly.

Auto Loading of Plugins

- Support for auto-loading libraries when an unknown class is being referenced.
- The auto-loading mechanism reads the files $ROOTSYS/etc/system.rootmap, ~/.rootmap and ./.rootmap (via TEnv) to try to map the unknown class to a library.
- If the library is found it, and the libraries on which it depends, are loaded.
- The rootmap files are created with the rlibmap tool when executing "make map".
- Example: in an interactive session, one can do directly

```
TLorentzVector v;
```

without having to do

```
gSystem->Load(“libPhysics”);
```

SLAC's New File Server - xrootd

- The file server xrootd (eXtended ROOT daemon) has been developed by Andy Hanushevsky of SLAC.
- The server exploits a multithreaded architecture to provide high-performance file based access, focusing on scalability and fault tolerance. The server is being extensively used by the BaBar collaboration. For more see Andy's talk on Thursday afternoon.
- The xrootd file server will in the near future replace the current daemon rootd.

The New xrootd Client - TXNetFile

- The new client class TXNetFile implements the xrootd protocol and is provided to open a file via the xrootd daemon.
- TXNetFile can detect when it talks to on old rootd daemon and return a TNetFile.
- To open a file via xrootd, just use the standard static method TFile::Open() as for opening via rootd.

Many Improvements in TThread

- Added thread safe TThread::Printf(), TThread::Info(), Warning() and Error() methods. These last three behave like the global TError methods, i.e. they need as location the full “class::method”.
- TThread::Join() when called in the main thread does not lock anymore the program but spawns a JoinHelper thread while in the meanwhile the main thread keeps processing system events.
- General cleanup in the TThread class to streamline the under laying POSIX pthread and Win32 thread drivers.
- TThread classes are now also available on win32 (NT4.0 and above), with some limitations though.

XML Parser Interface

- TXMLParser is the base class
- TSAXParser parses XML files using the SAX (Simple API for XML) interface. SAX is an event driven parser.
- TDOMParser parses XML files using the DOM (Document Object Model) interface. DOM is a platform and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.
- The SAX and DOM parsers are internally using libxml2.

TImageDump

- Many extensions to the libAfterImage library to support line, marker and (filled) polygon drawing. Accessible via TASImage.
- The new class TImageDump uses TASImage and derives from TVirtualPS to allow the saving of canvases in gif, jpg, png, tiff, etc., image formats in **batch mode**:

```
$ root –b
root [0] .x hsimple.C
root [1] c1->Print("c1.gif");
```

- Or to display any gif, jpg, png, tiff in a canvas, do:

```
TCanvas *c1;
TImageDump *imgdump = new TImageDump("test.png");
c1->Paint();
imgdump->Close();
```

TMacro

- This class allows for storing a C++ macro in a ROOT file.
- In addition to being stored in a ROOT file a TMacro can be executed, edited, etc.

```
TMacro m("Peaks.C");      //macro m with name "Peaks" is created
                          //from file Peaks.C
m.Exec();                 //macro executed with default arguments
m.Exec("4");              //macro executed with argument
m.SaveSource("newPeaks.C");
TFile f("mymacros.root","recreate");
m.Write();                //macro saved to file with name "Peaks"
```

TFileMerger

- This new class allows for easy copying of two or more files using the many TFile plugins (i.e. it can copy from Castor to dCache, or from xrootd to Chirp, etc.).

```
TFileMerger m;
m->Cp("srcUrl", "destUrl");
or
m->AddFile("url1");
m->AddFile("url2");
m->Merge();
```

- The AddFile() and Merge() use the Cp() to copy the file locally before making the merge, and if the output file is remote the merged file will be copied back to the remote destination.

New TFile Feature

- Support for opening files in raw mode when the file url contains the option string "filetype=raw", like "anyfile.tgz?filetype=raw".
- This allows TFile and its many remote access plugins to be used to open and read any file.
- This is used by the TFileMerger::Cp() method to copy any file from and to Grid storage elements (e.g. from Castor to dCache, from xrootd to a local file, and all possible permutations).

Miscellaneous

- gcc 4.0.x is supported
- MacOS X Tiger on PowerPC and on Intel
- Improved rpm and debian packaging scripts
- Coding style rules checked nightly:
  - http://root.cern.ch/root/nightly/codecheck/codecheck.html
- Should we move to subversion?
  - History tracking of moved files/directories
  - Better authentication for multiple writers

Plans

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");
```

Automatic Plugin Cache Generation

- Using the new “rlibconfig” 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

Using PCRE for Reg Exp's

- We are going to introduce a new class TPRegexp which uses the Perl Compatible Regular Expressions library.
- Well know, rich, regular expression syntax.
- Will be interfaced to TString and other class and methods now using TRegexp.
- TRegexp will of course stay for backward compatibility.