An iAIDA primer

Version 1.0

Anton Lechner,
IT-PSS-ED,
CERN

March 12, 2008

1 Introduction

This document provides an introduction of how to install and use the iAIDA package, created and maintained by Andreas Pfeiffer. Basic information concerning the package can be found on the iAIDA webpage `http://iadia.dynalias.net/iAIDA.html`.

The aim of this document is to summarize the installation steps and to provide some simple, but practical example of how to use the iAIDA library once it is built. Detailed information concerning the installation and usage of iAIDA can be found in various sources. This document tries to combine this information in a simple manner, and furthermore attempts in providing additional hints.

2 Installation of iAIDA on SLC 4

The following steps provide some detailed guidance how to install iAIDA and Grace on SLC 4 workstations. The procedure is similar for other Red Hat Enterprise Linux (RHEL) or Fedora Core/Fedora based systems. The installation can also be easily adapted for other Linux distros, by using installation tools peculiar to that system.

The instructions are given for performing a central iAIDA installation on the system, in order to make the library available for all users who have an account on the workstation. Thus, for the installation procedure it is required to have root permissions. Furthermore, a network connection is obligatory to download the package.

The installation of Grace is not required for building and using iAIDA. However, if one wants to use Grace as plotting tool, Grace must be installed prior to the iAIDA build process. AIDA objects can be stored in formats compatible with many other plotting tools, and hence in case one does not want to use Grace, the subsection 2.1 may be skipped.

2.1 Instructions for installing Grace using yum

In order to install Grace (`http://plasma-gate.weizmann.ac.il/Grace`) on a SLC 4 workstation, one can utilize the Fedora Core (FC) 4 yum repository. Please note, that the installation from newer FC repositories ($\geq$5) will fail since SLC 4 is based on RHEL 4, and thus imcompatibilities occur.

Following procedure shows how to perform the Grace installation (root permissions are required):

1. The first step is to add "FC4-extras" as yum repository. To do so, login as root

   ```
   su
   ```

   (you are prompted for the root password) and change to the directory /etc/yum.repos.d

   ```
   cd /etc/yum.repos.d
   ```

   Create a file named "FC4.repo", which contains the following lines:

   ```
   [fc4-extras]
   name=Fedora Core 4 Extras
   mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=extras-4&arch=$basearch
   enabled=0
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-extras
   gpgcheck=0
   ```

2. Check if Grace is found in the repository by executing

   ```
   yum --enablerepo=fc4-extras search grace
   ```

   The output should contain a text line like

   ```
   grace.i386 5.1.20-1.fc4 fc4-extras
   ```

   followed by a small description of the package itself. If this is the case grace can be installed (still as root) by executing

```
yum --enablerepo=fc4-extras install grace
```

If everything works out fine, i.e. all dependencies can be resolved, you are prompted for a final decision to install grace: Just type 'y' and grace will be installed.
In order to use Grace with the iAIDA package, the Grace developement package must also be installed. This can be done by

```
yum --enablerepo=fc4-extras install grace-devel
```

2.2 Instructions for building and installing iAIDA

To build the iAIDA package from source and to install the library and header files following instructions can be followed:

1. Create a directory (e.g. in your home directory) for building the iAIDA library:

   ```
   mkdir analysis
   ```

2. Download the iAIDA package from the webpage `http://iadia.dynalias.net/iAIDA.html` into the created directory:

   ```
   cd analysis
   wget http://cern.ch/pfeiffer/iAIDA-1.0.11.tgz
   ```

3. Extract and dearchive the package:

   ```
   tar xzvf iAIDA-1.0.11.tgz
   ```

   NOTE: Useful information for building the package can be found in the `README` file contained in the extracted directory.

4. In order to build the iAIDA library on your system the `e2fsprogs`, `expat` and `boost` packages are required (including their development packages). Thus, check the basic dependencies:

   - `e2fsprogs` and `e2fsprogs-devel` (ext2 file system utilities):

     ```
     yum search e2fsprogs
     ```

     Among other text, the output of this command should contain lines similar to:

     ```
     e2fsprogs.i386 1.35-12.4.EL4 installed
     e2fsprogs-devel.i386 1.35-12.4.EL4 installed
     ```

     NOTE: The important expression is the last one, since it denotes that the package is installed on your system.

   - `expat` and `expat-devel` (XML parser package):

     ```
     yum search expat
     ```

     Among other text, the output of this command should contain lines similar to:

     ```
     expat.i386 1.95.7-4 installed
     expat-devel.i386 1.95.7-4 installed
     ```

   - `boost` and `boost-devel` (nice C++ library package):

     ```
     yum search boost
     ```

     Among other text, the output of this command should contain lines similar to:

     ```
     expat.i386 1.95.7-4 installed
     expat-devel.i386 1.95.7-4 installed
     ```

NOTE: If you want to use iAIDA to create HBOOK or ROOT files you also need to have the CERNLIB and ROOT packages installed (the installation of these packages is not covered here).

5. If all required packages were found on the system, one can start with the building procedure. Change to the extracted iAIDA directory

```
cd iAIDA-1.0.11
```

and execute

```
./configure
```

Note that some arguments might be required to successfully configure the installation setup:

- The ./configure script tries to find the required libraries and developement packages (expat and boost) as well as optional libraries and packages (Grace, CERNLIB, ROOT) in the standard installation directories. If any of these packages was installed in an unusual location, the ./configure must be run with appropriate arguments, like `--with-cernlib=<cernlib installation directory>`. Execute

  ```
  ./configure --help
  ```

  for more information on the various possible arguments.
  NOTE: The CERNLIB package is normally preinstalled on SLC 4, and is located in the directory `/cern/pro`. Thus run the configure with the option `--with-cernlib=/cern/pro`.

- The ./configure script creates a setup containing the required path names of packages, and the installation directory, which is in the current case `/usr/local`. The installation directory can be changed by using the argument `--prefix=<installation directory>` when executing `configure`.

For example, `./configure` might be run as follows

```
./configure --prefix=/usr --with-cernlib=/cern/pro
```

(from the output you can see if the various packages were found). Once the makefile was created the iAIDA library can be built:

```
make
```

6. If the building process was successful the libary (and header files) can be installed. In order to do so, root permissions are required:

```
su
```

(you are prompted for the root password). Then execute:

```
make install
```

The iAida library is now installed in the `/usr/local/lib` directory (or in the `lib` subdirectory of the directory which was specified as prefix in step 5), and the header files are stored in the AIDA subdirectory in `/usr/local/include` (or in the `include` subdirectory of the directory which was specified as prefix in step 5).

3 Adjusting the makefile

In order to use the iAIDA package for your application, it is required to adjust the makefile by including the correct include and library paths, as well as the library name.

To facilitate this, the `configure` script, used in the iAIDA installation procedure, created a python script called `aida-config`, which can be found in `/usr/local/bin` (or in the `bin` subdirectory of your individual installation directory). This script can be used to generate strings suitable as `g++` compiler arguments:

- To get the required include pathes for using the iAIDA library, execute:
  ```
  aida-config --include
  ```
- To get the required library path and name, excute:
  ```
  aida-config --lib
  ```

The output of the script can be directly included into the makefile of your application by using quotes, i.e. ‘`aida-config --include`‘ or ‘`aida-config --lib`‘.

NOTE: If the execution of `aida-config` fails with the error message “command not found”, then the path of the directory containing the `aida-config` script is not included in your `PATH` environment variable. To include it, execute (in bash)

```
export PATH="/usr/local/bin:$PATH"
```

or (in csh/tcsh)

```
setenv PATH "/usr/local/bin:$PATH"
```

In case you used a different installation directory, above arguments must be modified accordingly. For convenience you can include the appropriate line in the start-up script for your shell.

4 iAIDA in practice - some guidelines

4.1 Basic program structure

The following simple example illustrates the basice use of iAIDA to create a histogram, which is, after being filled with a single value, stored in a XML file (compliant with the AIDA DTD). The comments explain the various arguments that can be given when creating a new tree (and assocting a store) and a new histogram. NOTE: As a header one needs to include `AIDA.hh`.

```
AIDA::IAnalysisFactory* analysisFactory  = AIDA_createAnalysisFactory();

AIDA::ITreeFactory* treeFactory = analysisFactory -> createTreeFactory()

AIDA::ITree* tree = treeFactory -> create("test.xml",    // file name
                                          "xml",         // file format
                                          false,         // file is not read-only
                                          true,          // a new file is created
                                          "uncompressed" // no compression is used
                                          );
delete treeFactory;

AIDA::IHistogramFactory* histogramFactory =
                           analysisFactory -> createHistogramFactory(*tree);

AIDA::IHistogram1D* aidaObject =
            histogramFactory -> createHistogram1D("MyHistogram", // histogram name
                                                  2,   // number of bins
                                                  0.0, // lower histogram boundary
```

```
                                                    2.0  // upper histogram boundary
                                                    );
delete histogramFactory;

aidaObject -> fill(0.5);

tree -> commit();
tree -> close();

delete tree;
delete analysisFactory;
```

The program structure can be summarized as follows: An analysis factory must be created, which in turn is used to create a tree factory. The tree factory allows to create a tree, that can be associated to a store (e.g. memory, file). As a further step, the analysis factory is utilized to create an instance of a histogram factory, where the (pointer to the) tree (the histogram should belong to) must be speciefied as argument. The histogram factory allows the user to create a range of analysis objects (e.g. histograms, clouds,...). For simplicity, above example only deals with a single histogram. The various analysis objects can be filled by using the appropriate member functions. Once all objects are filled, the tree must be committed (if associated to a file the objects are now written to the file) and closed.

In order to avoid memory leaks, the user has to delete the following objects: `treeFactory`, `histogramFactory`, `tree` and `analysisFactory` (as shown in above example). The `aidaObject` object must not be deleted by the user, since this is automatically done when deleting the tree. The `ITreeFactory` instance can be deleted immediately after creating the tree, and the `IHistogramFactory` instance after creating the histogram. The `ITree` and `IAnalysisInstances` are only deleted after commiting and closing the tree.