HOW TO Mantid – R. Leal – September 2014 – (c) ILL

1 Starting

1.1 Cloning the repo

http://www.mantidproject.org/Category:Development

```
mkdir $HOME/git
cd $HOME/git
git clone git@github.com:mantidproject/mantid.git
```

Mantid will be cloned to : `$HOME/git/mantid`

1.2 Dependencies

In Ubuntu, use apt-get to install the dependencies:

```
sudo apt-get  install git cmake-qt-gui libboost-all-dev libpoco-
dev libssl-dev libmuparser-dev libgsl0-dev libnexus0-dev qt4-
default qt4-qmake qt4-dev-tools libqwt5-qt4-dev libqwtplot3d-qt4-
dev liboce-visualization-dev libqscintilla2-dev libgoogle-
perftools-dev python-scipy python-qt4-dev python-sip python-sip-
dev ipython-qtconsole doxygen python-sphinx texlive texlive-latex-
extra dvipng
```

Download NeXus Library 4.3 and compile it from source:

http://download.nexusformat.org/kits/

Make sure the configure command will enable hdf4 and hdf5 support. Some extra dependencies, not mentioned above, might be need (????).

1.3 Compiling the source code

Once the code is cloned, run cmake:

```
cd $HOME/git/mantid
mkdir $HOME/git/mantid/Build
```

1.3.1 Without Paraview:

```
cmake -G "Eclipse CDT4 - Unix Makefiles" \
-DCMAKE_ECLIPSE_GENERATE_SOURCE_PROJECT=TRUE
-D_ECLIPSE_VERSION=4.3.1 \
-DCMAKE_ECLIPSE_MAKE_ARGUMENTS=-j8 -DCMAKE_BUILD_TYPE=Debug \
../Code/Mantid
make -j8
```

1.3.2 With Paraview

Download Paraview source code from here:
http://download.mantidproject.org/

Install in `/opt/ParaView-3.98.1-source` .

Then build Paraview :

```
cd /opt/ParaView-3.98.1-source
sudo mkdir Build
cd Build
## Don't know if this is still needed!
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/ParaView-3.98.1-
source/Build/lib
sudo cmake \
-DBUILD_TESTING=OFF \
-DBUILD_SHARED_LIBS=ON \
..
sudo make -j8
```

Then compile Mantid:

```
cd $HOME/git/mantid/Build
cmake -G "Eclipse CDT4 - Unix Makefiles" \
-DCMAKE_ECLIPSE_GENERATE_SOURCE_PROJECT=TRUE
-D_ECLIPSE_VERSION=4.3.1 \
-DCMAKE_ECLIPSE_MAKE_ARGUMENTS=-j8 -DCMAKE_BUILD_TYPE=Debug \
-DMAKE_VATES=TRUE -DParaView_DIR=/opt/ParaView-3.98.1-source/Build
\
../Code/Mantid
make -j8
```

1.4 Creating Loaders

Example below shows an example to create a loader called LoadPSI

```
cd ~/git/Mantid/Code/Mantid/Build
./class_maker.py --alg DataHandling LoadPSI
```

Files created:

```
      ../Framework/DataHandling/inc/MantidDataHandling/LoadPSI.h
      ../Framework/DataHandling/src/LoadPSI.cpp
      ../Framework/DataHandling/test/LoadPSITest.h
```

1.5 Creating Algorithms

Example below shows an example to create an algorithm called DetectorEfficiencyCorUser

```
cd ~/git/Mantid/Code/Mantid/Build
./class_maker.py --alg Algorithms DetectorEfficiencyCorUser
```

Files created:

```
      ../Framework/Algorithms/inc/MantidAlgorithms/DetectorEfficien
cyCorUser.h
      ../Framework/Algorithms/src/DetectorEfficiencyCorUser.cpp
      ../Framework/Algorithms/test/DetectorEfficiencyCorUserTest.h
```

2 Loaders

The aim of a Loader is to build a workspace with the data included in the data file and to assign an instrument definition to the workspace previously created. The instrument is created through an instrument definition file (IDF). See below.

For the ILL the workspaces are usually of type Workspace2D.

In general, every loader has the method:

```
 int confidence(Kernel::NexusDescriptor & descriptor) const
```

It returns a value as high as the confidence of the Loader for a certain instrument. It must not do any time demanding operations. For the nexus files, for example, it only checks if some paths are or are not available for a certain instrument.

There is an auxiliary class named `LoadHelper` with methods useful for the ILL Loaders.

All ILL Loaders start with `LoadILL*`.

2.1 Instrument Definition

To add a new instrument, the new instrument must be added to the file:

```
Code/Mantid/instrument/Facilities.xml
```

Under the XML tag.

`<facility name="ILL" (...)>`

In addition, a new IDF and an optional parameter file must be created following the convention:

`<instrument name>_Definition.xml`

`<instrument name>_Parameters.xml`

The files must be saved in the folder:

`Code/Mantid/instrument/`

Instructions to create these files are available here:

http://www.mantidproject.org/InstrumentDefinitionFile

http://www.mantidproject.org/Create_an_IDF

As well as an example for a SANS instrument:

http://www.mantidproject.org/IDF-ISIS-SANS2D-annotated

2.2 Workspace creation

The method below creates a workspace2D :

```
ws = WorkspaceFactory::Instance().create("Workspace2D", nVectors, xLength,
yLenght);
```

nVectors : number of histograms (number of detectors) + number of monitors.

xLenght : Number of time bins = number of channels + 1

yLenght : Number of channels

2.3 Units

For time of flight (TOF) instruments, in order to use the majority of the algorithms available in Mantid, the workspace must have TOF as the base unit.

At the ILL, the base units are channel number. Software, such as LAMP, converts the channel number into Energy. In Mantid, the data must be loaded in TOF, and then, existing algorithms (e.g. `ConvertUnits`), can convert from TOF to the desired Units.

The loaders must, therefore, load the data into TOF. First, one must creat the `theoreticalElasticTOF` which is the TOF from the source to the detector. Then, the `calculatedDetectorElasticPeakPosition` (the channel number where the elastic peak lays) must be found. Later, the time bins array - `tofBins` - must be set.

```
for (size_t i = 0; i < numberOfChannels + 1; ++i) {
  tofBins[i] = theoreticalElasticTOF + channelWidth
    * (i – calculatedDetectorElasticPeakPosition) - channelWidth / 2;
}
```

2.4 Properties

A property is a pair key – value. Some properties must exist assigned to the workspace for some algorithms to run correctly. For instance, the incident energy (`Ei`) is needed by the `ConvertUnits` algorithm. Properties are set in the workspace `run` object as the following:

```
API::Run & runDetails = workspace->mutableRun();
runDetails.addProperty<double>("Ei", energyValue, true);
```

In addition, there is a function (**`addNexusFieldsToWsRun`**) that automatically reads all the NeXus file fields and creates the respective workspace Properties. The function is defined in the`LoadHelper`. See, for example, **`loadNexusEntriesIntoProperties`** in **`LoadILLReflectometry`** file for an usage example.

2.5 Moving parts

Some loaders position the detectors according to motor positions (e.g. SANS).

If moving a single positional parameter, the best option would be performing this operation trough a child algorithm named `MoveInstrumentComponent`. See **`moveDetectorDistance`** in `LoadILLSANS`.
`LoadHelper` class also provides additional methods to get the current position of a component (**`getComponentPosition`**) and to move (**`moveComponent`**) or rotate a component (**`rotateComponent`**).

3 Coding

3.1 Git

Git workflow is here:

http://www.mantidproject.org/Git_Workflow

See the Setup section for the git mantid macros.

3.2 Build servers

Jenkins build servers are here:

http://builds.mantidproject.org/

After the code was pushed to GitHub, don't forget to keep an eye on the Static analysis tab:

http://builds.mantidproject.org/view/Static%20Analysis/

4 System Tests

When tests execute in more than 2 seconds they must be integrated in the system tests repository and they can not be part of the mantid framework. For IO tests (Loading & Saving) the threshold is more generous (> 5 seconds). The UnitTests are written in Python and also live in GitHub.

Information can be found online:

http://www.mantidproject.org/System_Tests

4.1 Repository

The repository can be cloned as previously explained. The address is the following:

`git@github.com:mantidproject/systemtests.git`

4.2 Structure

The ILL data files loaded by the Load tests are located in the folder:
`Data/ILL/`

The code it self responsible or testing algorithms is in
`SystemTests/AnalysisTests/*.py`
The ILL tests start with the prefix `ILL`.

To run a test type:
`StressTests/ runStressTests.py -R <test file pattern>`
You might need to set in advance the MANTIDPATH and the PYTHONHOME for running the script. Type `StressTests/ runStressTests.py -h` for more details.