MeqTrees on Opensuse 10.3 (64-bit)

This shows all the steps that I took to install successfully MeqTrees on my OpenSuSe 10.3 64-bit desktop.

Useful links

The Meqtrees Wiki page http://www.astron.nl/meqwiki/ , in particular the links pointing to "Building Timba" (http://www.astron.nl/meqwiki/BuildingTimba/ActualBuild)

In the Wiki, there are several further links concerning the building of all software requirements for building MeqTrees.

Major Problems and Observations

SuSe tends to use lib64/ subdirectories rather than lib/ for all 64-bit stuff. This seems to cause a lot of aggravation when building Timba. I have included all the generation of the required symbolic links.

Several commands (make, make installs etc) require root privileges. Occasionally, using sudo is not enough and you have to login explicitely as root. (or with root access).

PyQt is broken in SuSe and cannot be removed etc and substituted with something working (problems with Scintialla). I have forced the rebuilding of PyQt (otherwise MeqTrees would not work). It may have some side effects, however.

Step 1. Getting Timba

I am registered with Astron, hence I used Subversion + ssh to check yout (later to update) Timba:

svn co svn+ssh://salvini@lofar9.astron.nl/var/svn/repos/trunk/Timba

It will prompt for the password etc. It is also possible to check out as anonymous (see the Wiki page).

Software Requirements

The following table shows all the software requirements to build Meqtrees:

Mandatory	
Timba	The core package
Casacore	Usig casacore instead of AIPS++ libraries is highly recommended
Pyrap	Python casacore interface
PyQt	Python Qt bindings etc (source of trouble in OS 10.3)
Blitz	C++ scientific library

Qwt nd PyQwt	To produce graphs etc in meqbrowser
Optional	
Casarest	To produce images from measurements sets
karma	To visualize the images

Step 2. Building Casacore

First, download casacore from http://www.google.com/casacore

Have a good look also at the MeqTrees Wiki page http://www.astron.nl/meqwiki/LinkingWithCasaCore

These are the Casacore requirements:

- scons
- cfitsio
- wcslib
- lapack/blas
- fftw3
- flex/bison
- fortran compiler and fortran-to-c library, e.g. gfortran and libgfortran

On my system, this is what needed to do:

scons

It was already available (in SuSe repository anyway)

cfitsio

Download from http://heasarc.nasa.gov/fitsio/fitsio.html then:

```
tar xvzf cfitsio3090.tar.gz
cd cfitsio/
./configure --prefix=/usr/local   ("./configure --help"  gives a list of available options)
sudo make install
```

Because of the use of lib and lib64 in OS 10.3, you should now make a symbolic link

```
ln -s /usr/local/lib/libcfitsio.a  /usr/local/lib64/libcfitsio.a
```

wcslib

Download it from http://www.atnf.csiro.au/pub/software/wcslib/wcslib.tar.gz

```
tar xvzf wcslib.tar.gz
cd wcslib-4.3/
```

```
./configure --prefix=/usr/local
make install
cp wcsconfig.h /usr/local/include/wcslib
```

Painfully, the lib vs. lib64 issue bites you again. See how we build Casacore later on)

lapack, blas

Available in OS 10.3. Can install from Yast etc (need lapack, liblapack3, blas, libblas3). I have not tried to use more efficient BLAS & LAPACK such as Intel MKL library etc.

fftw3

Available in OS 10.3. Can install from Yast etc (need fftw3 and fftw3-devel)

Flex and bison

Already available in OS 10.3 (or installable from repositories)

gfortran

Available in my system, but can be easily installed from repository.

Finally, build casacore!

```
bzip2 -d casacore-0.3.0patched.tar.bz2
tar xvf casacore-0.3.0patched.tar
cd casacore-0.3.0/
./batchbuild.py enable_hdf5=false prefix=/usr/local \
           wcsincdir=/usr/local/include extrafflags=-fPIC \
           wcslibdir=/usr/local/lib64/ install -j3
```

Use the –j3 only if you are compiling on a multicore system! (Otherwise omit –j 3)

Step 3. Pyrap

First, download casacore from http://code.google.com/p/pyrap

Have a good look also at the MeqTrees Wiki page

These are the pyrap requirements:

- python
- numpy
- numarray

python

It was already available (in SuSe repository anyway)

numpy

Available from repository (Yast) (python-numpy and python-numpy-debug)

numarray 1.5.2

Download from http://sourceforge.net/project/showfiles.php?group_id=1369&package_id=32367 then:

```
cd numarray-1.5.2/
python setup.py install
ln -s /usr/local/include/python2.5/numarray/ /usr/include/python2.5/
```

Finally, build pyrap:

```
cd /usr/lib64/python2.5/site-packages/numpy/core/include/numpy
ln -s numpyconfig.h config.h
cd pyrap_fitting && ln -s tags/pyrap_fitting-0.1.0/ current \
./batchbuild.py numpyincdir=/usr/lib64/python2.5/site-packages/numpy/core/include\
                casacoreroot=/usr/local
```

Step 3. Build PyQt

The original PyQt in OS 10.3 is not quite suitable as it gives trouble with QScintilla.

These are the PyQt requirements:

- PyQt 3.x
- QScintilla 1

Both can be downloaded from http://www.riverbankcomputing.co.uk/software/pyqt/download3

On my system, this is what needed to do:

qscintilla 1

```
tar xvzf QScintilla-1.71-gpl-1.7.1.tar.gz
cd QScintilla-1.71-gpl-1.7.1/
cd qt/
/usr/lib/qt3/bin/qmake qscintilla.pro
make
```
(This may have to be run from within a shell with root privileges)

pyqt 3.x

```
tar xvzf PyQt-x11-gpl-3.17.4.tar.gz
cd PyQt-x11-gpl-3.17.4
python configure.py -n /usr/lib/qt3/include -o /usr/lib/qt3/lib
make -j4
make install
```

Step 4. Build Blitz++

First, download from http://sourceforge.net/projects/blitz/

Look also at the Wiki page on Blitz
http://www.astron.nl/meqwiki/BuildingTimba/RequiredSoftware/BlitzBuildNotes)

Finally, carry out the following:

```
tar xvzf blitz-0.9.tar.gz
cd blitz-0.9/
export CC=gcc
export CXX=g++
export FC=gfortran
export F77=gfortran
export F90=gfortran
./configure --enable-64bit --enable-shared --enable-optimize \
            --prefix=/usr/local/blitz/gnu4/ CXX="g++ -fPIC"
make
make install
cd /usr/local/blitz
sudo ln -s /usr/local/blitz/gnu4/lib /usr/local/blitz/gnu4/lib64
```

Step 5. qwt and pyqwt

These are required to get some graphis into the MeqTrees browser.

Download PyQwt-4.2.3 from http://pyqwt.sourceforge.net/. This includes also Qwt 4.2.0 which is all we need. Hence with root privileges (**IMPORTANT!**):

```
tar xvzf PyQwt-4.2.3.tar.gz
cd PyQwt-4.2.3/qwt-4.2.0/
/usr/lib/qt3/bin/qmake qwt.pro
make
install -m 644 include/* /usr/local/include
install -m 755 lib/libqwt.so.4.2 lib/libqwt.a /usr/local/lib
ldconfig
```

There may be problems here: what I did to sort them out was (possibly you need to be root):

```
cd /usr/local/lib
chmod 755 libqwt.so.4.2.0
ln -s libqwt.so.4.2.0 libqwt.so.4.2
ln -s libqwt.so.4.2.0 libqwt.so.4
ln -s libqwt.so.4.2.0 libqwt.so
```

Now complete PyQwt

`cd ../configure` (i.e.: cd ~/PyQwt-4.2.3/configure)

```
python configure.py -Q ../qwt-4.2.0
make
make install
```

Step 6. Build Timba at last!

This is actually simple. First, however, you have to make an appropriate variant file in ~/Timba/autoconf_share. The file must be called variant.*host* where *host* is the name of the system where you are building Timba. Have a look at the variants available and modify them accordingly. Here

is the variant file entry I added for using gcc 4.x on my OpenSue box (it is only a small portion of the variant file)

```
gnu4.compiler.conf: CXX=ccache\ g++ --enable-shared --with-cppflags="-msse \
       -m3dnow -msse2 -Wno-deprecated " --with-threads --with-sse \
       --with-ldflags="--enable-new-dtags"
gnu4.compiler.aipspp.var: --with-casacore=/usr/local --with-wcs=/usr/local
```

```
cd ~/Timba
./bootstrap
mkdir -p build/gnu4_debug
cd build/gnu4_debug
../../lofarconf
make -j 4
```

Optional Step 7. Try to run Timba

You could check if everything hangs together:

```
cd ~/Timba/install
source timba-init.sh
./make-symlink-tree.sh symlinked-gnu4_debug
cd ~
_timba-setup symlinked-gnu4_debug
meqbrowser.py
```

You should then be able to start the browser and from there the server, load a TDL etc

Optional Step 8. Making Images

For that, you could use casarest and karma, for example.

Casarest

This is included in the Timba download. So the procedure is quite simple:

```
cd ~/Timba/casarest
./batchbuild.py prefix=/usr/local casacoreroot=/usr/local install
```

You can now test lwimager:

```
cd image
lwimager ms=demo.MS
```

karma

To visualise images etc you could download karma from http://www.atnf.csiro.au/computing/software/karma/index.html using the following procedure

```
rsync -a rsync.science-software.net::karma/common/ /usr/local/karm
 /usr/local/karma/csh_script/install-karma
```

You can now view the image

```
cd ~/image
/usr/local/karma/bin/kvis demo-I-mfs1.fits
```