Bayeux/trunk installation report on (X)Ubuntu 14.04 LTS (64bits)

François Mauger, LPC Caen <mauger@lpccaen.in2p3.fr>

2016-03-06

In this document we propose an installation procedure for the Bayeux/trunk library on top of Cadfaelbrew (2016.01) on Xubuntu 14.04 LTS system (64bits). The build is done using the C++98 standard.

The target system

Architecture:

```
$ uname -a
Linux mauger-laptop 3.13.0-74-generic #118-Ubuntu SMP ...
```

Processors:

```
$ cat /proc/cpuinfo | grep "model name"
model name	: Intel(R) Core(TM) i7-3540M CPU @ 3.00GHz
model name	: Intel(R) Core(TM) i7-3540M CPU @ 3.00GHz
model name	: Intel(R) Core(TM) i7-3540M CPU @ 3.00GHz
model name	: Intel(R) Core(TM) i7-3540M CPU @ 3.00GHz
```

Linux version:

```
$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.3 LTS"
```

Installation of Cadfaelbrew

Links:

- Cadfaelbrew repository (GitHub)
- Cadfael (SuperNEMO Wiki)

Please follow the instructions on the installation report at https://nemo.lpc-caen.in2p3.fr/browser/Bayeux/trunk/doc/InstallationReports/Cadfaelbrew/Xubuntu14.04-a/tagged/cadfaelbrew_xubuntu14.04-a_report-0.1.pdf

Once you have installed Cadfaelbrew, you should be able to run a *brew* session:

```
$ brewsh
```

It opens a new shell with all environmental variables activated to setup all the software tools managed through Cadfaelbrew.
Alternatively you can use a dedicated setup function:

```
$ do_cadfaelbrew_setup
NOTICE: Cadfaelbrew is now setup !
```

You can check the location and version of core software utilities:

```
$ which cmake
/path/to/Cadfaelbrew/install/supernemo/cxx11/Cadfael.git/bin/cmake
```

```
$ cmake --version
cmake version 3.4.0
```

```
$ g++ --version
g++ (Homebrew gcc49 4.9.2_2) 4.9.2
```

```
$ doxygen --version
1.8.10
```

Ninja is a build system which can be used in place of (GNU)make. Install Ninja through `brew` if it was not already done before (you must setup the brew environment for that):

```
$ brewsh
$ brew install ninja
```

You can check your Ninja version:

```
$ ninja --version
1.6.0
$ exit
```

Installation of Bayeux (trunk)

Install dependencies:

```
$ sudo apt-get install gnuplot gnuplot-doc gnuplot-mode
$ sudo apt-get install libqt4-dev libqt4-dev-bin libqt4-gui
$ sudo apt-get install libreadline-dev readline-common
$ sudo apt-get install pandoc pandoc-data
$ sudo apt-get install python-docutils
```

Set the software base directory where there is enough storage capacity to host Bayeux (> 1 GB). Here we use a simple environment variable `SW_WORK_DIR` which points to a specific directory on the filesystem:

```
$ export SW_WORK_DIR=/data/sw
```

You should adapt this base directory to your own system, for example:

```
$ export SW_WORK_DIR=${HOME}/Software
```

Then create a few working directories:

```
$ mkdir -p ${SW_WORK_DIR}
$ mkdir ${SW_WORK_DIR}/Bayeux          # base working directory for Bayeux
$ mkdir ${SW_WORK_DIR}/Bayeux/Source   # hosts the source code
$ mkdir ${SW_WORK_DIR}/Bayeux/Binary   # hosts the build/installation directories
```

Download Bayeux/trunk source files:

```
$ cd ${SW_WORK_DIR}/Bayeux/Source
$ svn co https://nemo.lpc-caen.in2p3.fr/svn/Bayeux/trunk Bayeux-trunk
$ cd Bayeux-trunk
$ LANG=C svn info
Path: .
Working Copy Root Path: /data/sw/Bayeux/Source/Bayeux-trunk
URL: https://nemo.lpc-caen.in2p3.fr/svn/Bayeux/trunk
Relative URL: ^/Bayeux/trunk
Repository Root: https://nemo.lpc-caen.in2p3.fr/svn
Repository UUID: 3e0f96b8-c9f3-44f3-abf0-77131c94f4b4
Revision: 17214
Node Kind: directory
Schedule: normal
Last Changed Author: mauger
Last Changed Rev: 17210
Last Changed Date: 2016-03-04 23:36:04 +0100 (Fri, 04 Mar 2016)
```

Configure Bayeux:

1. Make sure Cadfaelbrew is setup on your system. If you follow the Cadfaelbrew installation report available from the Cadfael wiki page, you just have to invoke:

```
$ brewsh
```

or :

```
$ do_cadfaelbrew_setup
```

2. Create a build directory and cd in it:

```
$ mkdir -p ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Build-gcc-ninja-Linux-x86_64
$ cd ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Build-gcc-ninja-Linux-x86_64
```

3. Configure the Bayeux build with CMake and using Ninja and GCC :

```
$ cmake \
   -DCMAKE_BUILD_TYPE:STRING=Release \
   -DCMAKE_INSTALL_PREFIX:PATH=\
     ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64 \
   -DBAYEUX_CXX_STANDARD="98" \
   -DBAYEUX_COMPILER_ERROR_ON_WARNING=OFF \
   -DBAYEUX_WITH_IWYU_CHECK=ON \
   -DBAYEUX_WITH_DOCS=ON \
   -DBAYEUX_WITH_DOCS_OCD=ON \
   -DBAYEUX_WITH_DEVELOPER_TOOLS=ON \
   -DBAYEUX_WITH_EXAMPLES=ON \
   -DBAYEUX_WITH_BRIO=ON \
   -DBAYEUX_WITH_CUTS=ON \
   -DBAYEUX_WITH_MYGSL=ON \
   -DBAYEUX_WITH_DPP=ON \
   -DBAYEUX_WITH_MATERIALS=ON \
   -DBAYEUX_WITH_GEOMTOOLS=ON \
   -DBAYEUX_WITH_EMFIELD=ON \
   -DBAYEUX_WITH_GENVTX=ON \
   -DBAYEUX_WITH_GENBB_HELP=ON \
   -DBAYEUX_WITH_MCTOOLS=ON \
   -DBAYEUX_WITH_LAHAGUE=ON \
   -DBAYEUX_WITH_GEANT4_MODULE=ON \
   -DBAYEUX_WITH_MCNP_MODULE=OFF \
   -DBAYEUX_ENABLE_TESTING=ON \
   -GNinja \
   ${SW_WORK_DIR}/Bayeux/Source/Bayeux-trunk
```

Build (using 4 processors to go faster):

```
$ time ninja -j4
...
real 12m6.886s
user 43m4.932s
sys  2m24.929s
```

Quick check after build

After the build step, Bayeux uses the following hierarchy on the file system:

```
$ LANG=C tree -L 1 BuildProducts/
BuildProducts/
|-- bin/
|-- include/
|-- lib/
`-- share/
```

Particularly, the shared libraries are:

```
$ LANG=C tree -F BuildProducts/lib/
BuildProducts/lib/
|-- cmake/
|   `-- Bayeux-2.1.0/
```

```
|       |-- BayeuxConfig.cmake
|       |-- BayeuxConfigVersion.cmake
|       |-- BayeuxDocs.cmake
|       `-- BayeuxTargets.cmake
|-- libBayeux.so*
`-- libBayeux_mctools_geant4.so*
```

Executable are in:

```
$ LANG=C tree -L 1 -F BuildProducts/bin/
BuildProducts/bin/
|-- bxdpp_processing*
|-- bxg4_production*
|-- bxgenbb_inspector*
|-- bxgenbb_mkskelcfg*
|-- bxgenvtx_mkskelcfg*
|-- bxgenvtx_production*
|-- bxgeomtools_inspector*
|-- bxgeomtools_mkskelcfg*
|-- bxmaterials_diagnose*
|-- bxmaterials_inspector*
|-- bxmctools_g4_mkskelcfg*
|-- bxocd_make_doc*
|-- bxocd_manual*
|-- bxocd_sort_classnames.py*
|-- bxquery*
`-- bxtests/
```

These directories and files will be copied in the installation directory.

Test programs

Before to do the final installation, we run the test programs:

```
$ ninja test
[1/1] Running tests...
Test project /data/sw/Bayeux/Binary/Bayeux-trunk/Build-gcc-ninja-Linux-x86_64
        Start   1: datatools-test_reflection_0
  1/303 Test   #1: datatools-test_reflection_0 .......   Passed    0.28 sec
...
303/303 Test #303: bayeux-test_bayeux ................   Passed    0.09 sec

100% tests passed, 0 tests failed out of 303

Total Test time (real) =  83.62 sec
```

Installation

Run:

```
$ ninja install
...
```

Check installation

Browse the installation directory:

```
$ LANG=C tree -L 3 -F \
  ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64
/data/sw/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64
|-- bin/
|   |-- bxdpp_processing*
```

```
|   |-- bxg4_production*
|   |-- bxgenbb_inspector*
|   |-- bxgenbb_mkskelcfg*
|   |-- bxgenvtx_mkskelcfg*
|   |-- bxgenvtx_production*
|   |-- bxgeomtools_inspector*
|   |-- bxgeomtools_mkskelcfg*
|   |-- bxmaterials_inspector*
|   |-- bxmctools_g4_mkskelcfg*
|   |-- bxocd_make_doc*
|   |-- bxocd_manual*
|   |-- bxocd_sort_classnames.py*
|   `-- bxquery*
|-- include/
|   `-- bayeux/
|       |-- bayeux.h
|       |-- bayeux_config.h
|       |-- brio/
|       |-- cuts/
|       |-- datatools/
|       |-- dpp/
|       |-- emfield/
|       |-- genbb_help/
|       |-- genvtx/
|       |-- geomtools/
|       |-- materials/
|       |-- mctools/
|       |-- mygsl/
|       |-- qt/
|       |-- reloc.h
|       `-- version.h
|-- lib/
|   |-- cmake/
|   |   `-- Bayeux-2.1.0/
|   |-- libBayeux.so
|   `-- libBayeux_mctools_geant4.so
`-- share/
    `-- Bayeux-2.1.0/
        |-- Documentation/
        |-- examples/
        `-- resources/
```

Suggestions for a Bash setup (see below):

1. Define convenient environmental variables:

```
$ export SW_WORK_DIR=/data/sw
$ export BAYEUX_INSTALL_DIR=${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86
```

2. The only configuration you need now is:

```
$ export PATH=${BAYEUX_INSTALL_DIR}/bin:${PATH}
```

There is no need to update the `LD_LIBRARY_PATH` environment variable because Bayeux uses RPATH. So you **should NOT** use the following:

```
$ export LD_LIBRARY_PATH=${BAYEUX_INSTALL_DIR}/lib:${LD_LIBRARY_PATH}
```

3. After setting `PATH` as shown above, you can check where some of the executable are installed:

```
$ which bxquery
/data/sw/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64/bin/bxquery
```

Check datatools' OCD tool:

```
$ which bxocd_manual
/data/sw/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64/bin/bxocd_manual
$ bxocd_manual --action list
List of registered class IDs :
cuts::accept_cut
cuts::and_cut
...
mygsl::histogram_pool
```

Check geometry tools; cd in the Bayeux/geomtools example #01:

```
$ cd ${SW_WORK_DIR}/Bayeux/Source/Bayeux-trunk/source/bxgeomtools/examples/ex01
$ export CONFIG_DIR=$(pwd)/config
$ bxgeomtools_inspector --manager-config config/manager.conf

  G E O M T O O L S    I N S P E C T O R
  Version 5.0.0

  Copyright (C) 2009-2015
  Francois Mauger, Xavier Garrido, Benoit Guillon,
  Ben Morgan and Arnaud Chapon

  immediate help: type "help"
  quit:           type "quit"
  support:        Gnuplot display
  support:        Root display from GDML

geomtools> help
...
geomtools> display --help
...
geomtools> display
...
geomtools> list_of_logicals
...
geomtools> display optical_module.model.log
...
geomtools> list_of_gids --with-category optical_module.gc
List of available GIDs :
  [2020:0.0] as 'optical_module.gc'       [2020:0.1] as 'optical_module.gc'
  [2020:1.0] as 'optical_module.gc'       [2020:1.1] as 'optical_module.gc'
geomtools> display [2020:0.1]

Press [Enter] to continue...

geomtools>  export_gdml bxgeomtools_test.gdml
GDML file 'bxgeomtools_test.gdml' has been generated !
geomtools> quit
```

Conclusion:

- No problem for compiling, running tests and examples.

Setup your environment for Bayeux

Here we explicitely *load/setup* the Bayeux environment from a Bash shell with a dedicated function defined in my `~/.bashrc` startup file:

```
# The base directory of all the software (convenient path variable):
export SW_WORK_DIR=/data/sw

# The Bayeux/trunk setup function:
function do_bayeux_trunk_setup()
```

```
{
  do_cadfaelbrew_setup # Automatically load the Cadfaelbrew dependency
  if [ -n "${BAYEUX_INSTALL_DIR}" ]; then
      echo "ERROR: Bayeux/trunk is already setup !" >&2
      return 1
  fi
  export BAYEUX_INSTALL_DIR=${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86
  export PATH=${BAYEUX_INSTALL_DIR}/bin:${PATH}
  echo "NOTICE: Bayeux/trunk is now setup !" >&2
  return;
}
export -f do_bayeux_trunk_setup

# Special alias:
alias do_bayeux_dev_setup="do_bayeux_trunk_setup"
```

When one wants to use pieces of software from Bayeux, one runs:

```
$ do_bayeux_dev_setup
```

Then all executable are usable from the Bayeux installation directory:

```
$ which bxocd_manual
...
$ which bxgeomtools_inspector
...
$ which bxg4_production
...
```

Update the source code from the Bayeux/trunk

1. Activate the Cadfaelbrew environment:

   ```
   $ do_cadfaelbrew_setup
   ```

   or enter a `brew` shell:

   ```
   $ brewsh
   ```

2. Cd in the Bayeux/trunk source directory:

   ```
   $ cd ${SW_WORK_DIR}/Bayeux/Source/Bayeux-trunk
   ```

3. Update the source code:

   ```
   $ svn up
   ```

4. Cd in the Bayeux/trunk build directory:

   ```
   $ cd ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Build-gcc-ninja-Linux-x86_64
   ```

5. You may need to clean the build directory:

   ```
   $ ninja -clean
   ```

   and even to completely delete it to rebuild from scratch:

   ```
   $ cd ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/
   $ rm -fr Build-gcc-ninja-Linux-x86_64
   $ mkdir Build-gcc-ninja-Linux-x86_64
   $ cd Build-gcc-ninja-Linux-x86_64
   ```

   then reconfigure (see above).

6. You may need to delete the install tree:

   ```
   $ rm -fr ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64
   ```

7. Rebuild, test and install:

   ```
   $ ninja -j4
   $ ninja test
   $ ninja install
   ```

Alternative: build Bayeux with GNU make

a. Build dir:

```
$ mkdir -p ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Build-gcc-gnumake-Linux-x86_64
$ cd ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Build-gcc-gnumake-Linux-x86_64
```

b. Configure Bayeux with CMake and GNU make (default build system):

```
$ cmake \
   -DCMAKE_BUILD_TYPE:STRING=Release \
   -DCMAKE_INSTALL_PREFIX:PATH=\
     ${SW_WORK_DIR}/Bayeux/Binary/Bayeux-trunk/Install-gcc-Linux-x86_64 \
   -DBAYEUX_CXX_STANDARD="98" \
   -DBAYEUX_COMPILER_ERROR_ON_WARNING=OFF \
   -DBAYEUX_WITH_IWYU_CHECK=ON \
   -DBAYEUX_WITH_DOCS=ON \
   -DBAYEUX_WITH_DOCS_OCD=ON \
   -DBAYEUX_WITH_DEVELOPER_TOOLS=ON \
   -DBAYEUX_WITH_EXAMPLES=ON \
   -DBAYEUX_WITH_BRIO=ON \
   -DBAYEUX_WITH_CUTS=ON \
   -DBAYEUX_WITH_MYGSL=ON \
   -DBAYEUX_WITH_DPP=ON \
   -DBAYEUX_WITH_MATERIALS=ON \
   -DBAYEUX_WITH_GEOMTOOLS=ON \
   -DBAYEUX_WITH_EMFIELD=ON \
   -DBAYEUX_WITH_GENVTX=ON \
   -DBAYEUX_WITH_GENBB_HELP=ON \
   -DBAYEUX_WITH_MCTOOLS=ON \
   -DBAYEUX_WITH_LAHAGUE=ON \
   -DBAYEUX_WITH_GEANT4_MODULE=ON \
   -DBAYEUX_WITH_MCNP_MODULE=OFF \
   -DBAYEUX_ENABLE_TESTING=ON \
   ${SW_WORK_DIR}/Bayeux/Source/Bayeux-trunk
```

c. Build, test and install:

```
$ time make -j4
...
$ make test
$ make install
```