Bayeux/3.0.0 installation report on (X)Ubuntu 16.04 LTS (64bits)

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

2017-04-12

In this document we propose an installation procedure for the Bayeux 3.0.0 library (release candidate) on top of CadfaelBrew on Xubuntu 16.04 LTS (Xenial Xerus) for a 64-bits system.

Notes:

- Cadfaelbrew is only supported on 64-bits systems.
- Two build systems are supported : GNU/make and Ninja, on top of which CMake is used to build Bayeux.

Contents

The target system

- Architecture:

```
$ uname -a
Linux mauger-laptop 4.4.0-72-generic #93-Ubuntu SMP Fri Mar 31 14:07:41 UTC 2017 x86_64 x86
```

- 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=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.2 LTS"
```

- Environment:

  The system must have a relatively *bare* environment. It means that even if a lot of software has been installed on the system (`/usr/bin`) or in some alternative locations (`/usr/local`, `/opt`...), you should be able to run a shell with a lightweight `PATH`, typically something like:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/s
```

  In principle, you should not have the `LD_LIBRARY_PATH` environmental variable set:

```
$ echo aaa${LD_LIBRARY_PATH}ZZZ
aaaZZZ
```

Important notice: You have to check carefully both environment variables above because it is frequent that some system administrators use them to setup by default some third party software. The keyword here is *by default* which means something you didn't ask for. Unfortunately, the excessive/improper usage of these environs (mostly `LD_LIBRARY_PATH`) may ends to conflict while building Cadfael and/or Bayeux.

- Dependencies:

  It is be mandatory (or maybe useful) to install additional system packages to properly build Bayeux and activate some of its features. This is documented below.

Setup of Cadfaelbrew

Links:

- Cadfaelbrew repository (GitHub, public access)
- Cadfael (SuperNEMO Wiki, private access)

Please follow the instructions on the installation report at https://nemo.lpc-caen.in2p3.fr/browser/Bayeux/branches/Bayeux-3.0.0/doc/InstallationReports/Cadfaelbrew/Xubuntu16.04-a/tagged/cadfaelbrew_xubuntu16.04_report-1.0.pdf

brew

Once you have installed Cadfaelbrew (here in the `/opt/sw/SuperNEMO-DBD/Cadfaelbrew` directory), you should be able to run a *brew* session:

```
$ echo $PATH
/opt/sw/SuperNEMO-DBD/Cadfaelbrew/bin:/usr/local/sbin:...
$ which brew
/opt/sw/SuperNEMO-DBD/Cadfaelbrew/bin/brew
```

Enter a *brew shell*:

```
$ brew sh
...
```

This opens a new shell with all environmental variables activated to setup all the software tools managed through Cadfaelbrew (utilities, compiler(s), Boost, Root, Geant4...).

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

```
$ which cmake
/opt/sw/SuperNEMO-DBD/Cadfaelbrew/bin/cmake
$ cmake --version
cmake version 3.6.1

$ which g++
/usr/bin/g++
$ g++ --version
g++ (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
...

$ which doxygen
/opt/sw/SuperNEMO-DBD/Cadfaelbrew/bin/doxygen
$ doxygen --version
1.8.11
```

Ninja

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):

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

Then you can check your Ninja version:

```
$ which ninja
/opt/sw/SuperNEMO-DBD/Cadfaelbrew/bin/ninja
$ ninja --version
1.7.1
$ exit
```

Qt5

Qt5 is used for the Qt-based GUI components implemented in Bayeux (optional component). For now we use the system install of Qt5 (5.5.1 on Ubuntu 16.04):

```
$ sudo apt-get install  \
       libqt5core5a \
       libqt5gui5 \
       libqt5svg5 \
       libqt5svg5-dev \
       libqt5widgets5 \
       qtbase5-dev \
       qtbase5-dev-tools \
       qt5-default
```

Brew is able to install a recent Qt5 (Qt5.6.0) but it seems to be broken within Bayeux. Please do not use it as long as it is not fixed.

Configuration and build of Bayeux/3.0.0

Links:

- Bayeux (SuperNEMO Wiki, private access)

System dependencies

Install dependencies and useful utilities:

```
$ sudo apt-get install gnuplot gnuplot-doc gnuplot-mode
$ sudo apt-get install libreadline-dev readline-common
$ sudo apt-get install pandoc pandoc-data
$ sudo apt-get install python-docutils rst2pdf
```

See above for Qt5 components.

Working directory

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=/opt/sw
```

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

Download Bayeux

Download Bayeux/3.0.0 source files:

```
$ export BX_PRO_SOURCE_BASE_DIR="${SW_WORK_DIR}/Bayeux/Source"
$ mkdir -p ${BX_PRO_SOURCE_BASE_DIR}/branches
$ cd ${BX_PRO_SOURCE_BASE_DIR}/branches
$ svn co https://nemo.lpc-caen.in2p3.fr/svn/Bayeux/branches/Bayeux-3.0.0 Bayeux-3.0.0
$ export BX_PRO_SOURCE_DIR=${BX_PRO_SOURCE_BASE_DIR}/branches/Bayeux-3.0.0
$ cd ${BX_PRO_SOURCE_DIR}
$ pwd
/opt/sw/Bayeux/Source/branches/Bayeux-3.0.0
$ LANG=C svn info
Path: .
```

```
Working Copy Root Path: /opt/sw/Bayeux/Source/branches/Bayeux-3.0.0
URL: https://nemo.lpc-caen.in2p3.fr/svn/Bayeux/branches/Bayeux-3.0.0
Relative URL: ^/Bayeux/branches/Bayeux-3.0.0
Repository Root: https://nemo.lpc-caen.in2p3.fr/svn
Repository UUID: 3e0f96b8-c9f3-44f3-abf0-77131c94f4b4
Revision: 18948
Node Kind: directory
Schedule: normal
Last Changed Author: mauger
Last Changed Rev: 18906
Last Changed Date: 2017-04-03 21:22:33 +0200 (lun., 03 avril 2017)
```

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:

```
$ brew sh
```

2. Create a build directory and cd in it:

```
$ export BX_PRO_BIN_DIR="${SW_WORK_DIR}/Bayeux/Binary/Bayeux-3.0.0"
$ export BX_PRO_BUILD_DIR=${BX_PRO_BIN_DIR}/Build-ninja-Linux-x86_64
$ mkdir -p ${BX_PRO_BUILD_DIR}
$ cd ${BX_PRO_BUILD_DIR}
$ pwd
/opt/sw/Bayeux/Binary/Bayeux-3.0.0/Build-ninja-Linux-x86_64
```

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

```
$ export CADFAELBREW_PREFIX=$(clhep-config --prefix | tr -d '"')
$ export BX_PRO_INSTALL_DIR="${BX_PRO_BIN_DIR}/Install-Linux-x86_64"
$ cmake \
 -DCMAKE_BUILD_TYPE:STRING="Release" \
 -DCMAKE_INSTALL_PREFIX:FILEPATH="${BX_PRO_INSTALL_DIR}" \
 -DCMAKE_PREFIX_PATH:PATH="${CADFAELBREW_PREFIX}" \
 -DBAYEUX_CXX_STANDARD="11" \
 -DBAYEUX_WITH_IWYU_CHECK=ON \
 -DBAYEUX_WITH_DEVELOPER_TOOLS=ON \
 -DBAYEUX_WITH_LAHAGUE=OFF \
 -DBAYEUX_WITH_GEANT4_MODULE=ON \
 -DBAYEUX_WITH_MCNP_MODULE=OFF \
 -DBAYEUX_WITH_QT_GUI=ON \
 -DBAYEUX_ENABLE_TESTING=ON \
 -DBAYEUX_WITH_DOCS=ON \
 -DBAYEUX_WITH_DOCS_OCD=ON \
 -GNinja \
 ${BX_PRO_SOURCE_DIR}
```

Build

Using 4 processors to go faster (depends on your machine):

```
$ time ninja -j4
...
real 17m7.958s
user 62m46.852s
sys  3m19.396s
```

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-3.0.0/
|       |-- BayeuxConfig.cmake
|       |-- BayeuxConfigVersion.cmake
|       |-- BayeuxDocs.cmake
|       `-- BayeuxTargets.cmake
|-- libBXCatch.a
|-- libBayeux.so -> libBayeux.so.3*
|-- libBayeux.so.3 -> libBayeux.so.3.0.0*
|-- libBayeux.so.3.0.0*
|-- libBayeux_mctools_geant4.so -> libBayeux_mctools_geant4.so.3*
|-- libBayeux_mctools_geant4.so.3 -> libBayeux_mctools_geant4.so.3.0.0*
`-- libBayeux_mctools_geant4.so.3.0.0*
```

Executable are in:

```
$ LANG=C tree -L 1 -F BuildProducts/bin/
BuildProducts/bin/
|-- bxdpp_processing -> bxdpp_processing-3.0.0*
|-- bxdpp_processing-3.0.0*
|-- bxextract_table_of_objects*
|-- bxg4_production -> bxg4_production-3.0.0*
|-- bxg4_production-3.0.0*
|-- bxg4_seeds -> bxg4_seeds-3.0.0*
|-- bxg4_seeds-3.0.0*
|-- bxgenbb_inspector -> bxgenbb_inspector-3.0.0*
|-- bxgenbb_inspector-3.0.0*
|-- bxgenbb_mkskelcfg*
|-- bxgenvtx_mkskelcfg*
|-- bxgenvtx_production -> bxgenvtx_production-3.0.0*
|-- bxgenvtx_production-3.0.0*
|-- bxgeomtools_inspector -> bxgeomtools_inspector-3.0.0*
|-- bxgeomtools_inspector-3.0.0*
|-- bxgeomtools_mkskelcfg*
|-- bxmaterials_diagnose -> bxmaterials_diagnose-3.0.0*
|-- bxmaterials_diagnose-3.0.0*
|-- bxmaterials_inspector -> bxmaterials_inspector-3.0.0*
|-- bxmaterials_inspector-3.0.0*
|-- bxmctools_g4_mkskelcfg*
|-- bxocd_make_doc*
|-- bxocd_manual -> bxocd_manual-3.0.0*
|-- bxocd_manual-3.0.0*
|-- bxocd_sort_classnames.py*
|-- bxquery -> bxquery-3.0.0*
|-- bxquery-3.0.0*
|-- bxtests/
|-- bxvariant_inspector -> bxvariant_inspector-3.0.0*
`-- bxvariant_inspector-3.0.0*
```

These directories and files will be copied in the installation directory (but `bxtests/` which contains test programs usable only at build stage).

Test programs

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

```
$ ninja test
[0/1] Running tests...
Test project /opt/sw/Bayeux/Binary/Bayeux-3.0.0/Build-ninja-Linux-x86_64
     Start   1: bxdatatools-test_datatools
  1/341 Test   #1: bxdatatools-test_datatools ........................   Passed    0.11 sec
...
     Start 341: bayeux-test_bayeux_nocatch
341/341 Test #341: bayeux-test_bayeux_nocatch ........................   Passed    0.11 sec

100% tests passed, 0 tests failed out of 341

Total Test time (real) =  97.94 sec
```

Installation

Run:

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

Check installation

Browse the installation directory:

```
$ LANG=C tree -L 3 -F ${BX_PRO_INSTALL_DIR}
|-- bin/
|   |-- bxdpp_processing -> bxdpp_processing-3.0.0*
...
|   |-- bxquery -> bxquery-3.0.0*
|   |-- bxquery-3.0.0*
|   |-- bxvariant_inspector -> bxvariant_inspector-3.0.0*
|   `-- bxvariant_inspector-3.0.0*
|-- include/
|   `-- bayeux/
|       |-- bayeux.h
|       |-- bayeux_config.h
|       |-- bayeux_init.h
|       |-- brio/
|       |-- cuts/
|       |-- datatools/
|       |-- detail/
|       |-- dpp/
|       |-- emfield/
|       |-- genbb_help/
|       |-- genvtx/
|       |-- geomtools/
|       |-- materials/
|       |-- mctools/
|       |-- mygsl/
|       |-- reloc.h
|       |-- resource.h
|       `-- version.h
|-- lib/
|   |-- cmake/
|   |   `-- Bayeux-3.0.0/
|   |-- libBayeux.so -> libBayeux.so.3
|   |-- libBayeux.so.3 -> libBayeux.so.3.0.0
|   |-- libBayeux.so.3.0.0
|   |-- libBayeux_mctools_geant4.so -> libBayeux_mctools_geant4.so.3
|   |-- libBayeux_mctools_geant4.so.3 -> libBayeux_mctools_geant4.so.3.0.0
|   `-- libBayeux_mctools_geant4.so.3.0.0
`-- share/
    `-- Bayeux-3.0.0/
        |-- Documentation/
        |-- examples/
        `-- resources/
```

Suggestions for a Bash setup

1. Define convenient environment variables:

```
$ export SW_WORK_DIR=/opt/sw
$ export BX_PRO_INSTALL_DIR=\
       "${SW_WORK_DIR}/Bayeux/Binary/Bayeux-3.0.0/Install-Linux-x86_64"
```

2. The only configuration you need now is:

```
$ export PATH=${BX_PRO_INSTALL_DIR}/bin:${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=${BX_PRO_INSTALL_DIR}/lib:${LD_LIBRARY_PATH}
```

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

```
$ which bxquery
/opt/sw/Bayeux/Binary/Bayeux-3.0.0/Install-Linux-x86_64/bin/bxquery
```

Check Bayeux/datatools' OCD tool (Object Configuration Documentation):

```
$ which bxocd_manual
/opt/sw/Bayeux/Binary/Bayeux-3.0.0/Install-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 /opt/sw/Bayeux/Binary/Bayeux-3.0.0/Install-Linux-x86_64/share/Bayeux-3.0.0/examples/ge
$ export CONFIG_DIR=$(pwd)/config
```

Run the geometry inspector:

```
$ bxgeomtools_inspector --manager-config ${CONFIG_DIR}/manager.conf

  G E O M T O O L S    I N S P E C T O R
  Version 5.1.0

  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>
```

Test session:

```
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
```

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=/opt/sw
export BXSW_BASE_DIR=${SW_WORK_DIR}/Bayeux
export BX_PRO_BIN_DIR=${BXSW_BASE_DIR}/Binary/Bayeux-3.0.0
export BX_PRO_BUILD_DIR=${BX_PRO_BIN_DIR}/Build-ninja-Linux-x86_64

# The Bayeux/3.0.0 setup function:
function do_bayeux_300_setup()
{
 if [ -z "${CADFAELBREW_INSTALL_DIR}" ]; then
   echo >&2 "[error] Cadfaelbrew is not setup !"
   return 1
 fi
 if [ -n "${BX_PRO_INSTALL_DIR}" ]; then
   echo >&2 "[error] Bayeux 3.0.0 is already setup !"
   return 1
 fi
 export BX_PRO_INSTALL_DIR=${BX_PRO_BIN_DIR}/Install-Linux-x86_64
 export PATH=${BX_PRO_INSTALL_DIR}/bin:${PATH}
 echo >&2 "[notice] Bayeux 3.0.0 is now setup !"
 return 0;
}
export -f do_bayeux_300_setup

# Special alias:
alias bayeux_pro_setup="do_bayeux_300_setup"
```

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

```
$ brew sh
$ bayeux_pro_setup
```

Then all executable are usable from the Bayeux installation directory:

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

as well as package management utilities (CMake scripts...).

Update the source code from the Bayeux/3.0.0

1. Activate the Cadfaelbrew environment:

   ```
   $ brew sh
   ```

2. Cd in the Bayeux/3.0.0 source directory:

   ```
   $ cd ${SW_WORK_DIR}/Bayeux/Source/Bayeux-3.0.0
   ```

3. Update the source code:

   ```
   $ svn up
   ```

4. Cd in the Bayeux/3.0.0 build directory:

   ```
   $ export BX_PRO_BIN_DIR="${SW_WORK_DIR}/Bayeux/Binary/Bayeux-3.0.0"
   $ cd ${BX_PRO_BIN_DIR}/Build-ninja-Linux-x86_64
   ```

5. You may need to clean the build directory:

   ```
   $ ninja clean
   ```

   and even to completely delete and rebuild it from scratch:

   ```
   $ cd ${BX_PRO_BIN_DIR}
   $ rm -fr Build-ninja-Linux-x86_64
   $ mkdir Build-ninja-Linux-x86_64
   $ cd Build-ninja-Linux-x86_64
   ```

   then reconfigure (see above).

6. You may need to delete the install tree:

   ```
   $ rm -fr ${BX_PRO_BIN_DIR}/Install-Linux-x86_64
   ```

7. Rebuild, test and (re)install:

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