Falaise/trunk installation report on (X)ubuntu 16.04 LTS (64bits)

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

2016-08-25

In this document we propose an installation procedure for the Falaise/trunk (pre 3.0) library on top of Bayeux/trunk (pre 3.0) and Cadfaelbrew (2016.08) on Xubuntu 14.04 LTS system.

Contents

The target system

- Architecture:

```
$ uname -a
Linux mauger-laptop 4.4.0-34-generic #53-Ubuntu SMP Wed Jul 27 16:06:39 UTC 2016 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 16.04.1 LTS"
```

- Links:
  - Cadfaelbrew repository (GitHub, public access)
  - Cadfael (SuperNEMO Wiki, private access)
  - Bayeux (SuperNEMO Wiki, private access)
  - Falaise (SuperNEMO Wiki, private access)

Setup of Cadfaelbrew and Bayeux/trunk

You must have installed a standalone Bayeux/trunk on top of Cadfaelbrew.

Once you have installed Cadfaelbrew and Bayeux, you should be able to setup Bayeux:

```
$ brewsh           # Enter a *brew shell*
$ bayeux_dev_setup # Activate Bayeux
```

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
```

```
$ g++ --version
 g++ (Ubuntu 5.4.0-6ubuntu1~16.04.1) 5.4.0 20160609
```

```
$ which bxquery
/opt/sw/Bayeux/Binary/Bayeux-trunk/Install-gcc-cxx11-Linux-x86_64/bin/bxquery
$ bxquery --version
3.0.0
```

Configuration and build of Falaise/trunk

Working directory

Set the software base directory where there is enough storage capacity to host Falaise (> 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}/Falaise         # base working directory for Falaise
$ mkdir -p ${SW_WORK_DIR}/Falaise/Binary  # hosts the build/installation directories
```

Download Falaise

Download Falaise/trunk source files:

```
$ export FL_SOURCE_BASE_DIR="${HOME}/Documents/Private/Software/NEMO/SuperNEMO/Falaise/Sour
$ export FL_DEV_SOURCE_DIR=${FL_SOURCE_BASE_DIR}/Falaise-trunk
$ mkdir -p ${FL_SOURCE_BASE_DIR}
$ cd ${FL_SOURCE_BASE_DIR}
$ svn co https://nemo.lpc-caen.in2p3.fr/svn/Falaise/trunk Falaise-trunk
$ cd Falaise-trunk
$ LANG=C svn info
Path: .
Working Copy Root Path: /home/mauger/Documents/Private/Software/NEMO/SuperNEMO/Falaise/Sour
URL: https://nemo.lpc-caen.in2p3.fr/svn/Falaise/trunk
Relative URL: ^/Falaise/trunk
Repository Root: https://nemo.lpc-caen.in2p3.fr/svn
Repository UUID: 3e0f96b8-c9f3-44f3-abf0-77131c94f4b4
Revision: 17994
Node Kind: directory
Schedule: normal
Last Changed Author: garrido
Last Changed Rev: 17988
Last Changed Date: 2016-07-07 17:41:56 +0200 (jeu., 07 juil. 2016)
```

Configure Falaise

1. Make sure Cadfaelbrew and Bayeux are setup on your system. If you follow the Cadfaelbrew and Bayeux installation reports available from the Bayeux wiki page, you just have to invoke:

```
$ brewsh
$ bayeux_dev_setup
```

2. Create a build directory and cd in it:

```
$ export FL_DEV_BIN_DIR="${SW_WORK_DIR}/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk"
$ export FL_DEV_BUILD_DIR=${FL_DEV_BIN_DIR}/Build-gcc-cxx11-ninja-Linux-x86_64
$ mkdir -p ${FL_DEV_BUILD_DIR}
$ cd ${FL_DEV_BUILD_DIR}
$ pwd
/opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Build-gcc-cxx11-ninja-Linux-x86_64
```

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

```
$ echo ${CADFAELBREW_INSTALL_DIR}
/opt/sw/SuperNEMO-DBD/Cadfaelbrew
$ bxquery --prefix
/opt/sw/Bayeux/Binary/Bayeux-trunk/Install-gcc-cxx11-Linux-x86_64

$ export FL_DEV_INSTALL_DIR="${FL_DEV_BIN_DIR}/Install-gcc-cxx11-Linux-x86_64"
$ cmake \
 -DCMAKE_BUILD_TYPE:STRING=Release \
 -DCMAKE_INSTALL_PREFIX:PATH="${FL_DEV_INSTALL_DIR}" \
 -DCMAKE_FIND_ROOT_PATH:PATH="$(bxquery --prefix);${CADFAELBREW_INSTALL_DIR}" \
 -DFALAISE_COMPILER_ERROR_ON_WARNING=ON \
 -DFALAISE_ENABLE_TESTING=ON \
 -DFALAISE_WITH_DOCS=ON \
 -DFALAISE_WITH_DEVELOPER_TOOLS=ON \
 -DBoost_DIR:PATH="${CADFAELBREW_PREFIX_DIR}/lib/cmake" \
 -GNinja \
 ${FL_DEV_SOURCE_DIR}
```

Build

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

```
$ time ninja -j4
...
real 5m5.523s
user 18m37.232s
sys  0m57.296s
```

Quick check after build

After the build step, Falaise 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/
|-- Falaise/
|   `-- modules/
|       |-- libFalaise_CAT.so*
|       |-- libFalaise_ChargedParticleTracking.so*
|       |-- libFalaise_EventBrowser.so*
|       |-- libFalaise_GammaClustering.so*
|       |-- libFalaise_GammaTracking.so*
|       |-- libFalaise_MockTrackerClusterizer.so*
|       |-- libFalaise_TrackFit.so*
|       |-- libFalaise_VisuToy.so*
|       |-- libGammaTracking.so*
|       |-- libThings2Root.so*
|       `-- libTrackFit.so*
|-- cmake/
|   `-- Falaise-3.0.0/
|       |-- FalaiseBayeux.cmake
|       |-- FalaiseConfig.cmake
|       |-- FalaiseConfigVersion.cmake
|       `-- FalaiseTargets.cmake
|-- libFLCatch.a
`-- libFalaise.so*
```

Executable are in:

```
$ LANG=C tree -L 1 -F BuildProducts/bin/
BuildProducts/bin/
|-- flquery
|-- flreconstruct*
|-- flsimulate*
|-- fltests/
`-- flvisualize*
```

These directories and files will be copied in the installation directoryy (but `fltests`).

Test programs

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

```
$ ninja test
[0/1] Running tests...
Test project /opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Build-gcc-cxx11-ninja-Linux
       Start  1: falaise-test_snemo_datamodel_timestamp
1/38 Test  #1: falaise-test_snemo_datamodel_timestamp ................ Passed    0.13 sec
...
38/38 Test #38: falaisevisutoyplugin-test_visu_toy_module ............ Passed    1.71 sec

100% tests passed, 0 tests failed out of 38

Total Test time (real) =  20.31 sec
```

Installation

Run:

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

Check installation

Browse the installation directory:

```
$ LANG=C tree -L 3 -F ${SW_WORK_DIR}/Falaise/Binary/Falaise-trunk/Install-gcc-Linux-x86_64
|-- bin/
|   |-- flquery*
|   |-- flreconstruct*
|   |-- flsimulate*
|   `-- flvisualize*
|-- include/
|   `-- falaise/
|       |-- TrackerPreClustering/
|       |-- bipo3/
|       |-- exitcodes.h
|       |-- falaise.h
|       |-- resource.h
|       |-- snemo/
|       `-- version.h
|-- lib/
|   |-- Falaise/
|   |   `-- modules/
|   |-- cmake/
|   |   `-- Falaise-3.0.0/
|   `-- libFalaise.so
`-- share/
    `-- Falaise-3.0.0/
        |-- Documentation/
        `-- resources/
```

Suggestions for a Bash setup (see below)

1. Define convenient environmental variables:

```
$ export SW_WORK_DIR=/opt/sw
$ export FL_DEV_INSTALL_DIR=\
"${SW_WORK_DIR}/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Install-gcc-cxx11-Linux-x86_64
```

2. The only configuration you need now is:

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

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

```
$ which flquery
/opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Install-gcc-cxx11-Linux-x86_64/bin/flque
```

Setup your environment for Falaise

Here we explicitely *load/setup* the Falaise 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 FLSW_BASE_DIR=${SW_WORK_DIR}/SuperNEMO-DBD/Falaise
export FL_DEV_BIN_DIR=${FLSW_BASE_DIR}/Binary/Falaise-trunk
export FL_DEV_BUILD_DIR=${FL_DEV_BIN_DIR}/Build-gcc-cxx11-ninja-Linux-x86_64

# The Falaise/trunk setup function:
function do_falaise_trunk_setup()
{
  if [ -z "${CADFAELBREW_INSTALL_DIR}" ]; then
    echo "ERROR: Cadfaelbrew is not setup ! Please run 'brewsh'!" >&2
    return 1
  fi
  if [ -z "${BX_DEV_INSTALL_DIR}" ]; then
    echo "ERROR: Bayeux/trunk is not setup ! Please run 'bayeux_dev_setup'!" >&2
    return 1
  fi
  if [ -n "${FL_DEV_INSTALL_DIR}" ]; then
    echo "ERROR: Falaise/trunk is already setup !" >&2
    return 1
  fi
  export FL_DEV_INSTALL_DIR=${FL_DEV_BIN_DIR}/Install-gcc-cxx11-Linux-x86_64
  export PATH=${FL_DEV_INSTALL_DIR}/bin:${PATH}
  echo "NOTICE: Falaise/trunk is now setup !" >&2
  return 0;
}
export -f do_falaise_trunk_setup

# Special alias:
alias falaise_dev_setup="do_falaise_trunk_setup"
```

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

```
$ brewsh
$ bayeux_dev_setup
$ falaise_dev_setup
```

Then all executable are usable from the Falaise installation directory:

```
$ which flsimulate
/opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Install-gcc-cxx11-Linux-x86_64/bin/flsim
$ which flreconstruct
/opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Install-gcc-cxx11-Linux-x86_64/bin/flrec
$ which flvisualize
/opt/sw/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk/Install-gcc-cxx11-Linux-x86_64/bin/flvis
```

Update the source code from the Falaise/trunk

1. Activate the Cadfaelbrew/Bayeux environment:

   ```
   $ brewsh
   $ bayeux_dev_setup
   ..
   ```

2. Cd in the Falaise/trunk source directory:

   ```
   $ cd ${HOME}/Documents/Software/NEMO/SuperNEMO/Falaise/Source/Falaise-trunk
   ```

3. Update the source code:

   ```
   $ svn up
   ```

4. Cd in the Falaise/trunk build directory:

   ```
   $ export FL_DEV_BIN_DIR="${SW_WORK_DIR}/SuperNEMO-DBD/Falaise/Binary/Falaise-trunk"
   $ cd ${FL_DEV_BIN_DIR}/Build-gcc-cxx11-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 ${FL_DEV_BIN_DIR}
   $ rm -fr Build-gcc-cxx11-ninja-Linux-x86_64
   $ mkdir Build-gcc-cxx11-ninja-Linux-x86_64
   $ cd Build-gcc-cxx11-ninja-Linux-x86_64
   ```

   then reconfigure (see above).

6. You may need to delete the install tree:

   ```
   $ rm -fr ${FL_DEV_BIN_DIR}/Install-gcc-cxx11-Linux-x86_64
   ```

7. Rebuild, test and install:

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