Building SASSY

Brenton

7 April 2022

1 Introduction

This is a step-by-step guide to building SASSY using code checked out of the SourceForge (SF) repository.

Note that not everything is in the SF repository and you will need to contact me for some parts. Also various things will need to be installed from your distribution's repository.

SASSY is not a single project. It is made up of a collection of separate, almost independent projects that each need to be built. The order of building is important as the projects may depend on earlier projects. Also scripts introduced for one project may be called upon in subsequent projects.

This is a working document. It will be extended as new projects are added, and modified if changes are found to be required for the build process.

On SourceForge there is a primary SASSY project[1], which has very little at the moment, and from there a set of subprojects that you can access. In each subproject go to its `Code` page, and typically a `trunk` page under that. The exact details will be provided below for each project. However these instructions do not require you to visit SF in a web browser since the projects can be downloaded from the command line.

Some of the projects on SourceForge have tar files of the source code. These are probably out of date and are not refreshed very often since no one seems interested in them. By using the Subversion[2] code instead you can get the latest code moments after it is committed.

Important If any step in these instructions does not operate as expected please contact me before attempting to proceed further.

---

[1] https://sourceforge.net/projects/ocratato-sassy/

[2] Subversion is the version control system used for SASSY. Installation instructions are provided below.

2 Preliminaries

2.1 Linux Distribution

These instructions are for the *Fedora* distribution. This is probably only important for the repository commands and package names. If you really need to use something other than *Fedora* please contact me.

2.2 Build Tools

We are using the *Autotools* set with the gcc compiler (mostly with its g++ invocation.) These can be installed as follows:

```
sudo dnf group install 'C Development Tools and Libraries'
sudo dnf group install 'Development Tools'
```

2.3 The SASSY Directory

You will need a directory that will be the root of SASSY. Create a directory with a name of your choosing. (I just use `dev` but you might want to use `sassy`.)

Typically the following should be sufficient, but you might want it elsewhere.

```
cd
mkdir sassy
```

Important The instructions here result in the programs and libraries being installed in subdirectories under this directory rather `/usr/local` as is normal. This avoids the need to use `sudo` or otherwise get root privileges. Some scripts may assume this location.

2.4 The SASSY Environment Variable

There is one very important environment variable that is assumed to exist by most scripts - `SASSY`. This should be the path to the directory created in the previous step. Add the following to your `$HOME/.bash_profile` file (with the actual path you selected)

```
SASSY=$HOME/sassy
export SASSY
```

Log out, and back in again, and confirm that

```
cd $SASSY
```

takes you to the correct directory.

For the remainder of these instructions `$SASSY` will be used for the directory path.

2.5 The Projects Directory

SASSY has a lot of subprojects and they are grouped into a directory. Create the `projects` directory:

```
cd $SASSY
mkdir projects
```

2.6 Subversion

Use your system to install Subversion. It also has extensive documentation if you want it.

```
sudo dnf install subversion
sudo dnf install subversion-api-docs
```

Subversion has many subcommands, such as checkout and update. Once you have installed a project you can get it up to the latest version with just

```
svn update
```

We are now ready to start getting and building the subprojects.

3 Common Facilities Infrastructure

This is a library of useful bits of code. There is also included in the project the *Common Development Infrastructure* which includes support for testing.

3.1 Fetch the Code

First ensure you are in the projects directory:

```
cd $SASSY/projects
```

Note The following commands are for those that are just building the system for testing and evaluation. If you are going to do development that will involve committing changes please contact me.

Checkout the code with the following command:

```
svn checkout svn://svn.code.sf.net/p/ocratato-sassy/cfi/code/trunk/cfi/ cfi
```

Note that it consists of `svn` and its `checkout` subcommand, followed by the URL to the SourceForge repository, and lastly the name of the directory to create under `$SASSY/projects`.

You should confirm that $SASSY/projects/cfi has files such as `README`, `configure.ac`, `Makefile.am`, etc. and a `src` directory containing subdirectories for its constituent parts. Feel free to peruse the code, especially if you are intending to assist with development.

3.2 Prerequisites

The cfi modules depend on XML and for this we use `libxml2`. Use your system to install the development version, for example:

```
sudo dnf install libxml2-devel
```

3.3 Initialise the Project

3.3.1 Script setup.sh

The repository does not include many files that are required to build the system. Fortunately there are tools to create these files and I have a script to run them all.

Contact me for a copy of `setup.sh`. Place it in `$SASSY/projects` (for use on other projects), and ensure that it is executable and then run it to set up the project. For example:

```
cp setup.sh $SASSY/projects
cd $SASSY/projects
chmod +x setup.sh
cd cfi
```

```
../setup.sh
```

3.3.2 Build Directory

We use VPATH builds as these keep the source from being cluttered up with object files, libraries, and all the other guff the build process creates.

Create a build directory:

```
cd $SASSY/projects/cfi
mkdir build
```

3.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

As mentioned previously we want the executables, libraries, etc. installed under `$SASSY`, hence we set a prefix for the configure step:

```
cd $SASSY/projects/cfi
cd build
../configure --prefix=$SASSY
```

3.5 Building

You can just use the make command, but I like to capture the output into a log file for closer examination if something goes wrong.

Create a small script and place it your bin directory:

```
cat > $HOME/bin/mk <<EOF
#!/bin/bash
make &> make.log
if [ $? -ne 0 ]
then
    vi make.log
fi
EOF
```

(Of course you can use something other than vi if you wish.)

To build all you need is

```
cd $SASSY/projects/cfi/build
mk
```

3.6 Testing

The CFI module includes quite a bit of testing. You should probably run it to ensure things are set up OK.

```
cd $SASSY/projects/cfi/build
make check
```

Note that one test is of the timeout handling of the test harness itself, so it may appear to freeze for about 10 seconds.

3.7 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/cfi/build
make install
```

This will create various directories under `$SASSY`, such as `bin`, `lib`, `include` and `share`. You should confirm that this has been done.

4 The RDF C++ Library

This project aims to wrap librdf, the Redland libraries, in C++. It provides librdfxx for access to RDF data stores.

4.1 Fetch the Code

First ensure you are in the projects directory:

```
cd $SASSY/projects
```

Checkout the code with the following command:

```
svn checkout svn://svn.code.sf.net/p/ocratato-sassy/rdfxx/code/trunk/rdfxx/ rdfxx
```

You should confirm that `$SASSY/projects/rdfxx` has the expected files.

4.2 Prerequisites

4.2.1 Redland

The primary dependency is the C RDF library, `librdf` which includes `librasqal` and `libraptor`. Use your system to install the development version:

```
sudo dnf install redland-devel
sudo dnf install redland-pgsql
```

4.2.2 PostgreSQL

The intention is to use PostgreSQL for storage of most RDF. So far this has not been important, but the rdfxx library requires it. You will need both the server and client.

Note *These notes are a bit thin. It would help if you could take careful notes of your steps so these notes can be made a bit more detailed.*

For the client side install postgresql package, and also its jdbc interface[3],

```
sudo dnf install postgresql postgresql-jdbc
```

For the server the details are a bit sketchy, but PostgreSQL has a lot of good documentation to help you get it working properly. For now we don't need much, but eventually it will be important.

On my system I have the database server on a different virtual machine, but this is not necessary.

Install postgresql-server.

```
sudo dnf install postgresql-server
```

[3]This should be confirmed as it might have come in with redland-pgsql.

This will try to use `/var/lib/pgsql/data` to store the database. In my case I wanted it elsewhere so I created a bind based mount from `/data/database` to `/var/lib/pgsql/data`.

I also found it necessary to use chcon to set the context for the new data directory, otherwise systemd is unable to start the database.

The database then needs to be initialised:

```
su - postgres
initdb -D /var/lib/pgsql/data
```

I also enabled other machines to connect by adding '*' as a listen address in `postgresql.conf`; enabled other users to connect by adding a line to `pg_hba.conf`; and enabled `postgresql service` in the firewall.

Confirmation To verify a setup that will work for our project you should be able to log in to the database as yourself using one of the following commands:

```
psql
psql -h <hostname>
```

The `psql` command starts a shell on the database. This can be exited with \q

4.3 Initialise the Project

4.3.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/rdfxx
../setup.sh
```

4.3.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/rdfxx
mkdir build
```

4.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

As mentioned above we want the executables, libraries, etc. installed under `$SASSY`, hence we set a prefix for the configure step:

```
cd $SASSY/projects/rdfxx
cd build
../configure --prefix=$SASSY
```

4.5 Building

You can just use the make command, but I like to capture the output into a log file for closer examination if something goes wrong, for example, using the script we created previously:

```
cd $SASSY/projects/rdfxx/build
mk
```

4.6 Ontologies

Create a directory for storing RDF files:

```
cd $SASSY
mkdir ontologies
```

4.7 Testing

In order to test rdfxx you will need the `sassy.xml` configuration file. Please contact me for a copy. You should keep a backup of this file as something seems to clobber it[4]. Install it in the share directory:

```
cp sassy.xml $SASSY/share/sassy/
```

Run the tests for rdfxx. This includes a test of connectivity to PostgreSQL.

```
cd $SASSY/projects/rdfxx/build
make check
```

4.8 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/rdfxx/build
make install
```

[4]I suspect that building CFI is the culprit.

5 The RDFGUI Program

This program allows you to view and edit RDF models. For a better description see its user manual.

5.1 Fetch the Code

First ensure you are in the projects directory.

```
cd $SASSY/projects
```

Checkout the code with the following command:

```
svn checkout svn://svn.code.sf.net/p/ocratato-sassy/rdfgui/code/trunk/rdfgui rdfgui
```

5.2 Prerequisites

You will need Qt5 including the SVG module which is usually a separate package.

```
sudo dnf install qt5-devel qt5-qtsvg-devel
```

You will also need the Graphviz package. Currently we only use the `dot` program but this might change if we can get some time to extend the visualisation component.

```
sudo dnf install graphviz
```

5.3 Initialise the Project

5.3.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/rdfgui
../setup.sh
```

5.3.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/rdfgui
mkdir build
```

5.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

```
cd $SASSY/projects/rdfgui
cd build
../configure --prefix=$SASSY
```

5.5 Building

You can just use the make command, but I like to capture the output into a log file for closer examination if something goes wrong, for example, using the `mk` script:

```
cd $SASSY/projects/rdfgui/build
mk
```

5.6 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/rdfgui/build
make install
```

Create a desktop file, `rdfgui.desktop` in '$HOME/.local/share/applications' with the following contents:

```
[Desktop Entry]
Comment=A program for displaying and manipulating RDF data.
Terminal=false
Name=RDF-Gui
Exec=/home/nfs/sassy/dev/bin/rdfgui
Type=Application
Icon=/home/nfs/sassy/dev/share/sassy/logo3.png
Categories=TextTools;
```

Adjust the Exec and Icon paths to suit your system. Contact me for a copy of the logo file.

5.7 Documentation

Create a directory for documentation and a subdir for manuals.

```
cd $SASSY
mkdir -p docs/manuals
```

Contact me for a copy of RDFGUI manual.

5.8 Ontologies

I have found it useful to construct a few RDF models for testing and general messing about. From the Catalogue tab of rdfgui enter a name for the model in the New Model section, select the type of staorage it will use, and press the `New` button. Fill in the details (don't forget the description) and add it to the catalogue.

5.8.1 Core Schema

Contact me for a copy of the schema that has the core RDF structures. Copy it into the ontologies directory and add it into the catalogue.

6 Natural Language Generator

This program takes an RDF grammar tree and creates an English sentence (or phrase).

6.1 Fetch the Code

First ensure you are in the projects directory:

```
cd $SASSY/projects
```

Checkout the code with the following command

```
svn checkout svn://svn.code.sf.net/p/ocratato-sassy/nlg/code/trunk nlg
```

Note that the command doesn't have "nlg" at the end of the URL. I messed up the upload on this one, but it doesn't matter much.

6.2 Prerequisites

None, apart from what has been already built and installed.

6.3 Initialise the Project

6.3.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/nlg
../setup.sh
```

6.3.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/nlg
mkdir build
```

6.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

```
cd $SASSY/projects/nlg
cd build
../configure --prefix=$SASSY
```

6.5 Building

Just use make, or the script from previous projects:

```
cd $SASSY/projects/nlg/build
mk
```

6.6 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/nlg/build
make install
```

6.7 Ontologies

Contact me for copies of the NLG ontologies.

Copy them into them into $SASSY/ontologies and use `rdfgui` to add them to your catalogue.

7 Document Generator

This takes an RDF description of a document and creates a Markdown version. The Pandoc program is then used convert the Markdown into PDF.

7.1 Fetch the Code

It hasn't been installed into SourceForge yet, so contact me if you want this module.

7.2 Prerequisites

While not technically a prerequisite that is necessary for the building of the document generator, the Pandoc program is needed to get anything useful.

Use your system to install Pandoc:

```
sudo dnf install pandoc
```

7.3 Initialise the Project

7.3.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/docgen
../setup.sh
```

7.3.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/docgen
mkdir build
```

7.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

```
cd $SASSY/projects/docgen
cd build
../configure --prefix=$SASSY
```

7.5 Building

Just use make, or the script from previous projects:

```
cd $SASSY/projects/docgen/build
mk
```

7.6 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/docgen/build
make install
```

7.7 Ontologies

Contact me for copies of the document ontologies.

Copy them into them into $SASSY/ontologies and use `rdfgui` to add them to your catalogue.

8 Reasoner

This project integrates the FaCT++ inference engine into SASSY.

8.1 Create Directory

For the reasoner the structure of the directory is a little different since we have to build the prerequisite library.

```
cd $SASSY/projects
mkdir -p fact++/rdf-iface
```

8.2 Prerequisites

You will need to get and build FaCT++.

Get the source zip file from `https://bitbucket.org/dtsarkov/factplusplus/downloads/` The file is `FaCTpp-src-v1.6.5.zip`.

The zip file contains the core library plus a lot of interfaces we don't need. Also it is set up for MAC OS and there is a missing include file in one of the source files. To fix all of this I have created a patch that can be applied.

Contact me for a copy of the patch, `fact.patch`.

```
cp FaCTpp-src-v1.6.5.zip $SASSY/projects/fact++
cp fact.patch $SASSY/projects/fact++
cd $SASSY/projects/fact++
unzip FaCTpp-src-v1.6.5.zip
patch -p0 < fact.patch
cd FaCT++-src-v1.6.5/src
cmake
make
```

8.3 Fetch the Code

It hasn't been installed into SourceForge yet, so contact me if you want this module.

8.4 Initialise the Project

8.4.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/fact++/rdf-iface
../../setup.sh
```

8.4.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/fact++/rdf-iface
mkdir build
```

8.5 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

```
cd $SASSY/projects/fact++/rdf-iface
cd build
../configure --prefix=$SASSY
```

8.6 Building

Just use make, or the script from previous projects:

```
cd $SASSY/projects/fact++/rdf-iface/build
mk
```

8.7 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/fact++/rdf-iface/build
make install
```

9 Rule Engine

Rule engines are effectively an interpreted language for modifying RDF models. RDF data is found using SPARQL and can be used to create new statements in the model.

At the time of writing this project is still under development, but it has been stored in SourceForge's repository.

9.1 Fetch the Code

First ensure you are in the projects directory:

```
cd $SASSY/projects
```

Checkout the code with the following command

```
svn checkout svn://svn.code.sf.net/p/ocratato-sassy/reng/code/trunk/rule-engine \
    rule-eng
```

9.2 Prerequisites

For this project we use a compiler compiler called SableCC. Specifically we need a version that has been modified to generate languages beside Java.

9.2.1 Java

The SableCC program is a Java program, so we need Java installed. It is probably already installed (we only need the headless version), but if not then use your system to install it.

```
sudo dnf install java
```

9.2.2 SableCC Altgen

Get the zip file from `http://www.mare.ee/indrek/sablecc/`. The file is `sablecc-3-beta.3.altgen.20041114.zip`[5]

We will create a project directory for SableCC so we don't end up with Java stuff in the Rule Engine project.

```
cd $SASSY/projects
mkdir sablecc
cd $HOME/Downloads
cp sablecc-3-beta.3.altgen.20041114.zip $SASSY/projects/sablecc/
cd $SASSY/projects/sablecc
unzip sablecc-3-beta.3.altgen.20041114.zip
```

[5] I seem to have built my version against an older version of SableCC-Altgen so there might be some updates coming.

No further action is required since the zip file contains a Java jar file of sablecc which will be referenced by the rule engine build.

9.3 Initialise the Project

9.3.1 Script setup.sh

Use the setup.sh script that you got for the CFI project to initialise the autotools files:

```
cd $SASSY/projects/rule-eng/
../setup.sh
```

9.3.2 Build Directory

Create a build directory:

```
cd $SASSY/projects/rule-eng/
mkdir build
```

9.4 Configuring

This checks the state of the system, that prerequisites are installed and creates the Makefiles and build subdirectories.

```
cd $SASSY/projects/rule-eng/
cd build
../configure --prefix=$SASSY
```

9.5 Building

Just use make, or the script from previous projects:

```
cd $SASSY/projects/rule-eng/build
mk
```

9.6 Installing

To copy the programs, libraries, include files, etc. to their proper places;

```
cd $SASSY/projects/rule-eng/build
make install
```

10 View Your Mind (VYM)

This is an external program for doing mind maps. It is being investigated as a possible front end for SASSY.

10.1 Fetch the Code

10.1.1 Create Project

First create a project directory:

```
cd $SASSY/projects
mkdir vym
```

10.1.2 Download

Get the code from `https://github.com/insilmaril/vym/tree/maintained` where you will see a `Code` menu. Get the zip file[6].

Copy the zip file into $SASSY/projects/vim and run unzip on it.

10.2 Building

Build he project by cd'ing into the unzipped directory then

```
cmake
make
make install
make clean
```

This will install it into /usr/bin which is not ideal for SASSY, but will do for now.

10.3 Installation

Create desktop file, `vym.desktop` containing

```
[Desktop Entry]
Comment=A program for creating mind maps.
Terminal=false
Name=VYM
Exec=/usr/bin/vym
Type=Application
MimeType=application/x-vym
Icon=//usr/share/vym/icons/vym.png
Categories=Utility;TextTools;
```

[6]I should modify my build to use the `maintained` version rather than the `develop` that seems to be the default.

and copy it into $HOME/.local/share/applications. This will create an entry for it in the system menu.

11 Things To Be Done

This is some notes to me about things that need be fixed for the build process.

11.1 Tools, Configuration and Scripts

Scripts such as setup.sh, svn-check.sh should be committed to the SASSY master project. Similarly with sassy.xml.

11.2 Ontologies

Add the schema ontologies into version control. Add test ontologies into version control for the related projects.

11.3 Projects into Version Control

11.3.1 Docgen

The docgen program and associated scripts needs to be added to SourceForge.

11.3.2 Reasoner

The reasoner interface code and the fact.patch file need to be added to SourceForge

11.4 Documentation

Convert docs to Markdown format and add to SourceForge version control.

Papers that have been downloaded should be listed in a bibliography so that others can download them.