KELL Installation for Ubuntu16.04

Weikun Han

1. Install dependencies:

KLEE requires all the dependencies of LLVM, which are discussed here. In particular, you should install the following programs and libraries, listed below as Ubuntu packages [1]:

```
$ sudo apt-get install build-essential curl libcap-dev git cmake libncurses5-dev python-minimal python-pip unzip zlib1g-dev libboost-all-dev perl python flex bison
$ gcc --version
```

You will need gcc/g++ 4.8 or later installed on your system. For the virtual machine, you need increase the memory (4GB) to make the installation, and you need least 30GB hard disk space.

2. Install LLVM 3.4:

KLEE is built on top of LLVM; the first steps are to get a working LLVM installation. See Getting Started with the LLVM System for more information.

Here, I want to install it manually. First, I download llvm-3.4.src.tar.gz and clang-3.4.src.tar.gz. Create folder and name it as Projects in path /home/weikun. Creat folder and name it as llvm-3.4 in path /home/weikun/Projects. Create folders and name it as src and build in path /home/weikun/Projects/llvm-3.4.

```
$ cd
$ mkdir Projects
$ cd Projects
$ mkdir llvm-3.4
$ cd llvm-3.4
$ mkdir src
$ mkdir build
```

Next, unzip llvm-3.4.src.tar.gz and clang-3.4.src.tar.gz. Put all files in llvm-3.4 into path /home/weikun/Projects/llvm-3.4/src. Put all files in clang-3.4 into path /home/weikun/Projects/llvm-3.4/src/tools/clang (you need create clang folder).

Finally, locate in the path /home/weikun/Projects/llvm-3.4/build.

```
$ cd
$ cd /home/weikun/Projects/llvm-3.4/build
```

For the **release** version, do it.

```
$ /home/weikun/Projects/llvm-3.4/src/configure --prefix=/home/weikun/Projects/llvm-3.4/install --enable-assertions
```

For the **debug** version, do it.

```
$ /home/weikun/Projects/llvm-3.4/src/configure --prefix=/home/weikun/Projects/llvm-3.4/install -disable-optimized --enable-assertions
```

To make installation, do it.

```
$ make
$ make install
$ sudo gedit ~/.profile
```

Here, find the following line:

```
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
```

and change it to:

```
PATH="$HOME/bin:$HOME/.local/bin:$PATH:/home/weikun/Projects/llvm-3.4/install/bin"
```

Run the below command in terminal.

```
export PATH=$PATH:/home/weikun/Projects/llvm-3.4/install/bin
```

3. Install minisat:
KLEE supports multiple different constraint solvers. You must install at least one to build KLEE.

```
$ cd
$ cd Projects
$ mkdir minisat
$ cd minisat
$ mkdir build
$ mkdir src
```

I use git to download minisat. And then put all files in minisat into path /home/weikun/Projects/minisat/src.

```
$ cd
$ git clone https://github.com/stp/minisat.git
```

Finally, locate in the path /home/weikun/Projects/minisat/build.

```
$ cd
$ cd /home/weikun/Projects/minisat/build
```

For the **release** version, do it.

```
$ cmake -DCMAKE_INSTALL_PREFIX=/home/weikun/Projects/minisat/install/ ../src
```

For the **debug** version, do it.

```
$ cmake -DCMAKE_INSTALL_PREFIX=/home/weikun/Projects/minisat/install/ -
DCMAKE_BUILD_TYPE=Debug ../src
```

To make installation, do it.

```
$ make
$ make install
```

4. Install constraint solver STP:
STP Historically KLEE was built around only this solver so support for this solver is the most stable. KLEE is based on the STP constraint solver. STP does not make frequent releases, and its GitHub repository is under constant development and may be unstable. The instructions below are for the release 2.1.2.

```
$ cd
$ cd Projects
$ mkdir stp
$ cd stp
$ mkdir build
$ mkdir src
```

I use git to download stp. And then put all files in stp into path /home/weikun/Projects/stp/src.

```
$ cd
$ git clone https://github.com/stp/stp.git
```

Finally, locate in the path /home/weikun/Projects/minisat/build.

```
$ cd
$ cd /home/weikun/Projects/stp/build
```

For the **release** version, do it.

```
$ cmake -DBUILD_SHARED_LIBS:BOOL=OFF -DENABLE_PYTHON_INTERFACE:BOOL=OFF -
DCMAKE_INSTALL_PREFIX=/home/weikun/Projects/stp/install -
DMINISAT_INCLUDE_DIRS=/home/weikun/Projects/minisat/install/include -
DMINISAT_LIBDIR=/home/weikun/Projects/minisat/install/lib ../src
```

For the **debug** version, do it.

```
$ cmake -DBUILD_SHARED_LIBS:BOOL=OFF -DENABLE_PYTHON_INTERFACE:BOOL=OFF -
DCMAKE_INSTALL_PREFIX=/home/weikun/Projects/stp/install -
DCMAKE_BUILD_TYPE=Debug -
DMINISAT_INCLUDE_DIRS=/home/weikun/Projects/minisat/install/include -
DMINISAT_LIBDIR=/home/weikun/Projects/minisat/install/lib ../src
```

5. Install uclibc and the POSIX environment model:
By default, KLEE works on closed programs (programs that don't use any external code such as C library functions). However, if you want to use KLEE to run real programs you will want to enable the KLEE POSIX runtime, which is built on top of the uClibc C library.

```
$ cd
$ cd Projects
$ git clone https://github.com/klee/klee-uclibc.git
$ cd klee-uclibc
$ ./configure --make-llvm-lib
$ make
```

6. Install Google test sources:
For unit tests we use the Google test libraries. If you don't want to run the unit tests you can skip this step but you will need to pass -DENABLE_UNIT_TESTS=OFF to CMake when configuring KLEE in step 9. We depend on a version 1.7.0 right now so grab the sources for it.

```
$ cd
$ cd Projects
$ curl -OL https://github.com/google/googletest/archive/release-1.7.0.zip
$ unzip release-1.7.0.zip
$ rm -f release-1.7.0.zip
$ cd googletest-release-1.7.0
$ cmake .
$ make
```

7. Install KLEE:

KLEE must be built "out of source" so first make a binary build directory. You can create this where ever you like.

However, for debug version, KLEE can build in source. I use git to download klee source. First, I need move the libminisat.a file to the path /home/weikun/Projects/stp/build/lib.

```
$ cd
$ cd Projects
$ git clone https://github.com/klee/klee.git
$ cd klee
$ ./configure --with-stp=/home/weikun/Projects/stp/build --with-uclibc=/home/weikun/Projects/klee-uclibc --with-llvmsrc=/home/weikun/Projects/llvm-3.4/src --with-llvmobj=/home/weikun/Projects/llvm-3.4/build --with-llvmcc=/home/weikun/Projects/llvm-3.4/install/bin/clang --with-llvmcxx=/home/weikun/Projects/llvm-3.4/install/bin/clang++ --enable-posix-runtime
$ make CXXFLAGS='-std=c++11 -fexceptions' 2>a.txt
```

You're ready to go! Check the Tutorials page to try KLEE. And the debug version klee is in the path /home/weikun/Projects/klee/Debug+Asserts/bin/klee.

8. References:

[1] KLEE, https://klee.github.io/, March 2017.