27th August 2014

New approach to coverage analysis in RTEMS

In this post I present a new approach to coverage analysis in RTEMS. Old approach is described in my previous post: http://kmiesowicz.blogspot.com/2014/07/preparing-environment-for-rtems.html [http://kmiesowicz.blogspot.com/2014/07/preparing-environment-for-rtems.html]

Development of new tools for coverage analysis in RTEMS was the goal of my project in ESA SOCIS 2014. Old approach used a bunch of shell scripts to run coverage analysis. This project's goal was to add coverage analysis capabilities to rtems-test Python-based tool, created as a part of RTEMS Tools project.
Following this article you will be able to setup complete environment to perform coverage analysis for pc386 BSP for RTEMS.
[]

Rtems Source Builder

Rtems Source Builder is a tool for building packages from source. To not copy official documentation, I will redirect you to it: http://www.rtems.org/ftp/pub/rtems/people/chrisj/source-builder/source-builder.html [http://www.rtems.org/ftp/pub/rtems/people/chrisj/source-builder/source-builder.html] . This tool makes it much easier to get correct tools like compilers, debuggers, autotools etc. to work with RTEMS.
First, let's create development tree:

```
cd
mkdir development/rtems/src
cd development/rtems/src
```

Then let's clone rtems-source-builder repo.

```
git clone git://git.rtems.org/rtems-source-builder.git
cd rtems-source-builder
```

Then let's check if we have our environment correct.

```
source-builder/sb-check
```

You should see Environment OK. Otherwise, you will see which packages are needed - you have to install them. On Linux Mint 17 this command should help:

```
sudo apt-get build-dep binutils gcc g++ gdb unzip git python2.7-dev
```

Next you need to build proper toolset for our development target. I wanted to have toolset for i386 CPUs, BSP pc386 - to be able to run RTEMS within QEMU.

```
cd rtems ../source-builder/sb-set-builder --log=l-i386.txt --prefix=$HOME/development/rtems/4.11 4.11/rtems-i386
```

And voila! It should build everything correctly. In case of any problems you will see message with build log path - you have to open it and find what caused problem. It may be some missing libraries or dependencies which must be installed first.

Rtems

Next step is to get RTEMS source code.

```
cd
cd development/rtems/src
git clone git://git.rtems.org/rtems.git
```

Now, you should bootstrap, configure and build RTEMS.

Couverture-qemu

For running pc386 BSP standard QEMU version is enough. But if you want to perform coverage analysis, standard QEMU version will be insufficient. For this purpose you will need couverture-qemu. This section will go through build process for couverture-qemu. First, let's clone couverture-qemu repository.

```
cd
git clone https://scm.forge.open-do.org/git/couverture-qemu/couverture-qemu.git [https://scm.forge.open-do.org/git/couverture-qemu/couverture-qemu.git]
cd couverture-qemu/
git checkout qemu-stable
```

Then you will need a patch - or simply I needed it to get it working (special thanks to Dr Joel Sherril, who is an author of this patch). You can get this patch here: Download patch for couverture-qemu [http://goo.gl/osKnvU] After downloading the patch, you need to apply it with following command:

```
git apply couverture-qemu_patch.diff
```

Next you need to configure, make and install couverture-qemu:

```
./configure --prefix=$HOME/qemu/install --target-list=i386-softmmu --disable-virtfs --disable-werror --disable-docs
make
make install
```

Rtems-tools

Last part of environment is rtems-tools repo. This is the core of new approach. Let's clone the git repo:

```
cd
cd development/rtems/src
git clone git://git.rtems.org/rtems-tools.git
cd rtems-tools
```

After that you need to download, unzip and apply my patches [https://drive.google.com/file/d/0B79YDovTsA76NzNHN3BUTEZrbVE/edit?usp=sharing] - it's temporarily - until all my changes will finally be merged with mainline.
Next step is building covoar - c++ tool, which analyzes trace data, and performs actual coverage analysis. This temporarily requires cloning another repo into rtems-tools tree.

```
cd rtems-tools
git clone git://git.rtems.org/chrisj/rtl-host.git
```

Then you need to build the rld library, which will be then used by covoar built.

```
pushd rtl-host
waf configure build
popd
```

Now, we can build covoar

```
pushd tester/covoar
waf configure build
popd
```

Now, it's time to edit symbol sets configuration file. With new coverage module in rtems-test you can perform analysis for arbitrary set of libraries. Each symbol set must consists of at least one library, but single library can also be included in multiple sets. Symbol sets configuration file format is very simple:

```
symbolset:
 name = NAME_OF_SET
 lib = ABSOLUTE_PATH_OF_LIBRARY_1
 lib = ABSOLUTE_PATH_OF_LIBRARY_2
 ..
 ..
 ..
 lib = ABSOLUTE_PATH_OF_LIBRARY_N
symbolset:
 name = NAME_OF_ANOTHER_SET
 lib = ...
```

In rtems-tools/tester/rtems/testing/coverage/ there is exemplary symbol set config file (symbolSets.config), which contains multiple symbol sets. You need to adjust paths to built RTEMS libraries in this file, according to your RTEMS build directory path.

Last step is configuring your PATH, to point to all RTEMS tools, qemu and covoar:

```
export PATH=${HOME}/rtems-tools/tester/covoar/build/:${HOME}/development/rtems/4.11/bin:${HOME}/qemu/install/bin/:${PATH}
```

After this step you should be able to run your coverage analysis. Let's create special directory in $HOME, and perform the analysis there:

```
cd
mkdir coverage_test
cd coverage_test
```

```
$HOME/development/rtems/src/rtems-tools/tester/rtems-test --rtems-bsp=pc386 \
--log=log_pc386 --coverage \
--rtems-tools=/home/krzysztof/development/rtems/4.11 \
/home/krzysztof/development/rtems/src/b-pc386/i386-rtems4.11/c/pc386/testsuites/
```

You enables coverage analysis by adding --coverage option when invoking rtems-test. If you simply want to run tests, omit this option.

Now, you have time for a tea or coffee, because rtems-test will run all tests from testsuites directory (about 500 tests), and then perform multiple covoar runs - one for each symbol set specified in symbolSets.config file. So, keep calm and relax.

After everything is done (I hope it goes without errors), you should find report.html file which is the summary of coverage results for all symbol sets. From there you will be pointed to reports for every single symbol set. This partial reports are placed under test directory.

CAVEATS:

The coverage module is still under development, so it can change. I will update this post according to changes introduced. There are still areas of potential improvements, which will be announced on this blog. All comments, improvements or questions are welcome :-)

Opublikowano 27th August 2014, autor: KrzysiekM

0 Dodaj komentarz