BUILDING TESTING DEBUGGING PACKAGING

BUILDING OOREXX

René Vincent Jansen

27th International Rexx Language Symposium, Tampa 2016
AGENDA

- Getting the code
- Building
- Testing
- Debugging
- Packaging
GETTING THE CODE
GETTING THE CODE FROM SOURCEFORGE

- need: subversion (svn) client
- need: cmake
- need: make (or nmake on windows)
- need: ncurses

- [https://sourceforge.net/projects/oorexx/](https://sourceforge.net/projects/oorexx/)
  - here you can find where to point svn to:
    - `svn checkout svn://svn.code.sf.net/p/oorexx/code-0/main/trunk` oorexx-code-0
CMAKE
CMAKE

- modern form of autotools
  - a way to adapt C/C++ project builds to different platforms
- performs out-of-source builds
- make a build directory and cd into it
  - cmake path-to-source -options
- The whole build procedure (all platforms) is in the file CmakeLists.txt
CMAKE EXAMPLE

mkdir -p ../build

cd ../build

cmake -DBUILD_DEB=1 -DOS_DIST=ubuntu1604 -DCMAKE_BUILD_TYPE=RELEASE
$WORKSPACE

make clean  # make sure rexx picks up the current build date

make
MAKE
make

- cmake generates makefiles
- make is a build tool that (re)builds programs if the source is newer
- you can tell it about dependencies
- oldest and most standard build tool
- gnu make is nearly everywhere
Building on the Raspberry Pi

Raspbian Wheezy
The build needs cmake, at least GNU G++ 4.8.2 and the ncurses development library
The cmake in the raspbian wheeze distribution is too old; it needs to be built from source. Download and untar the 3.5.2 source package; then run ./bootstrap && make && make install - this will take care of make.
The C++ compiler on Wheezy is 4.6.3, it is too old and has severe bugs in template handling. From ooRexx 5.00 on, templates are required.
sudo apt-get install gcc-4.8-base sudo apt-get install g++-4.8
Finally, for a build the ncurses development header files are required. They can be installed like this:
sudo apt-get install libcurses5-dev
After this, do a standard cmake out-of-source build

Raspbian Jessie
sudo apt-get install cmake
After provisioning the virtual machine image:

```
sudo zypper install cmake
sudo zypper install ncurses-devel
```

and do a standard out-of-source cmake build.

Note that before a sudo make install, processes started using the rexx executable from the bin build directory do not disappear and need to be dispatched with kill -9. After installing rexx, this problem goes away. Note that in some virtual images there are problems involving the firewall and the rxapi daemon.
Prerequisites

- Subversion client (svn) from e.g. https://sourceforge.net/projects/win32svn/
- Cmake 3.2.3 from http://cmake.org

Probably down the line you will have to install NSIS, Xalan and Xerces, but the above is enough to build a local copy and run it. The ooRexx documentation is a different issue and needs other tools.

Environment variables

This set of environment variables is suggested; match this to your local environment

```
set TEST_DIR=C:\Users\rvjansen\oorexxttest
set SRC_DRV=C:\
set BLD_DIR=\Users\rvjansen\oorexxbuild
set REXX_BUILD_HOME=%SRC_DRV%\%BLD_DIR%
set REXX_HOME=%SRC_DRV%\%BLD_DIR%
call "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x64
set INCLUDE=%INCLUDE%;c:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\include;
path c:\NSIS;%REXX_BUILD_HOME%\bin;%PATH%;c:\Xalan\bin;c:\Xerces\bin;%TEST_DIR%;%TEST_DIR%\framework;
```
Check out the code from the Subversion repository
Checked out trunk to \Users\rvjansen\oorexx with:
svn co http://svn.code.sf.net/p/oorexx/code-0/main/trunk .
(if you want to commit stuff from here, you need the svn+ssh notation and your SF password)

Configure the build with cmake
Then switch to the \Users\rvjansen\oorexxbuild directory and issued:
cmake ..\oorexx -G "NMake Makefiles"
This tells cmake to generate the makefiles and not the default Visual Studio project. Important is to
clean out that directory every time something goes wrong, because cmake seems easy to confuse.
Most important here is that the compiler is happy in finding the include files and libraries it needs.

Run the build
Afterwards, in that same directory,
nmake
This builds the system in the bin directory of the oorexxbuild directory. It is runnable in that state.
TEST
the ooRexx source comes with its
  own testing tool (ooRexxUnit)
  own testing suite
run tests:
  rexx ./testOORexx.rex -s -X native_API -x socketClass
TEST SUITE

- ooRexxUnit is modeled on JUnit (and now xUnit)
- will finish a test suite and gives results afterwards
- this is useful because you have immediate insight in which classes pass and which classes fail
- after a source code update, all supported platforms should be tested immediately
DEBUG
BUGS

DEBUGGING

- make sure to build without `-DCBUILD_TYPE=Release`
- a build without this is non-optimized and has symbols for debugging
- you can use gdb to set breakpoints
- you can also add print statements
JENKINS
Automate your software builds
Distributed Master/Slave Model
Compatibility with existing systems/protocols
Build, deploy, test, report
Plugins for various environments
JENKINS TO Z/OS

- We run Jenkins from a Tomcat instance ... anywhere
- In this case an existing build server on Linux
- Jenkins is an easy tool
  - Master
  - Slaves
  - Credentials
  - Jobs
# Jenkins' Main Dashboard

This image shows the main dashboard of Jenkins, a popular open-source automation server. The dashboard includes a list of builds and their statuses, such as success or failure, along with the duration of the builds. The builds are categorized under various projects, each with its own build queue and executor status.

### Build Queue (1)
- ooRexx-Linux-Mint17

### Build Executor Status

<table>
<thead>
<tr>
<th>Executor</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMZ</td>
<td>1 idle</td>
</tr>
<tr>
<td>ams-01</td>
<td>1 idle</td>
</tr>
<tr>
<td>ams-02</td>
<td>1 idle</td>
</tr>
<tr>
<td>ams-03</td>
<td>offline</td>
</tr>
<tr>
<td>ams-04</td>
<td>1 idle</td>
</tr>
<tr>
<td>ide-01</td>
<td>offline</td>
</tr>
<tr>
<td>ubuntu16</td>
<td>1 idle</td>
</tr>
</tbody>
</table>

### Build List

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Success</th>
<th>Last Failure</th>
<th>Last Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>netrexx-commit</td>
<td>2 mo 2 days - #3</td>
<td>N/A</td>
<td>24 sec</td>
</tr>
<tr>
<td>netrexx-commit-Z</td>
<td>2 mo 1 day - #10</td>
<td>N/A</td>
<td>17 sec</td>
</tr>
<tr>
<td>ooRexx-Linux-Mint17</td>
<td>26 days - #24</td>
<td>1 mo 29 days - #12</td>
<td>2 min 51 sec</td>
</tr>
<tr>
<td>ooRexx-macOS-build</td>
<td>9 days 16 hr - #5</td>
<td>N/A</td>
<td>20 sec</td>
</tr>
<tr>
<td>ooRexx-macOS-test</td>
<td>9 days 16 hr - #54</td>
<td>9 days 16 hr - #53</td>
<td>3 min 58 sec</td>
</tr>
<tr>
<td>ooRexx-OpenSUSE-Tumbleweed-X86_64-build</td>
<td>9 days 4 hr - #48</td>
<td>19 days - #46</td>
<td>4 min 51 sec</td>
</tr>
<tr>
<td>ooRexx-OpenSUSE-Tumbleweed-X86_64-test</td>
<td>19 days - #15</td>
<td>9 days 4 hr - #19</td>
<td>3 min 9 sec</td>
</tr>
<tr>
<td>ooRexx-Raspbian-Jessie-build</td>
<td>9 days 16 hr - #59</td>
<td>1 mo 3 days - #51</td>
<td>9 min 56 sec</td>
</tr>
<tr>
<td>ooRexx-Raspbian-Jessie-test</td>
<td>9 days 16 hr - #66</td>
<td>9 days 23 hr - #65</td>
<td>5 min 1 sec</td>
</tr>
<tr>
<td>ooRexx-Raspbian-Wheezy-build</td>
<td>9 days 16 hr - #47</td>
<td>2 mo 0 days - #21</td>
<td>45 sec</td>
</tr>
<tr>
<td>ooRexx-Raspbian-Wheezy-test</td>
<td>9 days 16 hr - #13</td>
<td>9 days 22 hr - #12</td>
<td>7 min 28 sec</td>
</tr>
<tr>
<td>ooRexx-ubuntu16-build</td>
<td>9 days 15 hr - #27</td>
<td>N/A</td>
<td>2 min 21 sec</td>
</tr>
<tr>
<td>ooRexx-ubuntu16-test</td>
<td>9 days 15 hr - #28</td>
<td>9 days 22 hr - #27</td>
<td>13 min</td>
</tr>
<tr>
<td>ooRexx-Z-build</td>
<td>9 days 16 hr - #46</td>
<td>1 mo 3 days - #36</td>
<td>13 min</td>
</tr>
<tr>
<td>ooRexx-Z-test</td>
<td>9 days 16 hr - #28</td>
<td>9 days 23 hr - #27</td>
<td>3 min 1 sec</td>
</tr>
</tbody>
</table>
JENKINS

CONFIGURE A SLAVE LPAR
Configure a Job

Project name: netrex

Description:

Discard Old Builds

This build is parameterized

Disable Build (No new builds will be executed until the project is re-enabled.)

Execute concurrent builds if necessary

Restrict where this project can be run

Advanced Project Options

Source Code Management

- None
- CVS
- CVS Projectset
- Subversion

Modules

Repository URL: https://svn.kenai.com/svn/netrex/netrexxc-repo/netrexxc/trunk

Credentials: rvjansen/****** (kenai userid)
Specify what the job runs

Build

Execute shell

Command: make

See the list of available environment variables

Post-build Actions

Add build step

Add post-build action

Save  Apply
PUBLISH THE PACKAGES

- Install the “Publish over SSH” plugin
- Use credentials from Jenkins, not from slave machine
ANY QUESTIONS?

THANK YOU!