ROCK Linux

The Toolkit for Building Linux Distributions

Clifford Wolf - www.clifford.at

Introduction

LINBIT Information Technologies GmbH

The Problem

- Building a Linux distribution is complex and time-consuming
  - Even small distributions have a few hundret packages
  - Bootstrapping a distribution may take weeks when done cleanly
  - Crossbuilding can be very tricky
- Keeping it up-to-date is even more
  - All packages need to be monitored for updates
  - Updates must be tested and checked for regressions
  - Most regressions are in the depending packages
- All distibutions have most of the stuff in common
  - The packages themself are almost the same
  - Package configurations and init scripts are very similar

The Solution

- ROCK Linux makes distribution development easy:

```
./scripts/Config      -cfg  mydist
./scripts/Config      -cfg  myboot

./scripts/Download    -cfg  mydist   -required
./scripts/Download    -cfg  myboot   -required

./scripts/Build-Target      -cfg  mydist
./scripts/Build-Target      -cfg  myboot

./scripts/Create-ISO      demo  myboot   mydist
```

- Only the features specific to the new distribution need to be added.

- ROCK Linux makes easy things easy and xomplex things possible.

Config Tool

Abstract

- The ./scripts/Config tool is used to configure the to be built system
- The available config options are defined in config.in files
- The config.in files don't need to be specially registered with ./scripts/Config
- The config script is just a big shell script, the config.in files add additional calls to special functions to alter the config variables.

Call-Tree (1/2)

Execution of sub-scripts:

- achitecture/*/preconfig.in
* Selecting Architecture
* architecture/$ROCKCFG_ARCH/config.in

- misc/*/preconfig.in
- target/*/preconfig.in
- package/*/*/preconfig.in

* Selecting Target
* target/$ROCKCFG_TARGET/config.in

* misc/*/noexpertconfig.in

Only procedures marked with '*' might interact with the user (register options).

Call-Tree (2/2)

Execution of sub-scripts (continued):

* misc/*/noexpertconfig.in

* {package/*,misc}/*/config-*.in
* {package/*,misc}/*/config.in
* Various common build options

- package/*/*/postconfig.in
- misc/*/postconfig.in
- architecture/$ROCKCFG_ARCH/postconfig.in
- target/$ROCKCFG_TARGET/postconfig.in

Only procedures marked with '*' might interact with the user (register options).

Namespaces

Naming-scheme for extening config variables:

Core: ROCKCFG_*
Archs: ROCKCFG_ARCH_<Arch-Name>_*
Targets: ROCKCFG_TRG_<Target-Name>_*
Packages: ROCKCFG_PKG_<Pkg-Name>_*

Config-Internal Variables:

Core: CFGTEMP_*
Archs: CFGTEMP_ARCH_<Arch-Name>_*
Targets: CFGTEMP_TRG_<Target-Name>_*
Packages: CFGTEMP_PKG_<Pkg-Name>_*

Config Presets: ROCKCFGSET_*

Example config.in

<package/clifford/subversion/config.in>

```
if pkgcheck subversion X
then
    menu_begin MENU_PKG_SUBVERSION 'Subversion Options'
        bool 'Create a subversion-static package' \
                        ROCKCFG_PKG_SUBVERSION_STATIC 1
        if [ "$ROCKCFG_PKG_SUBVERSION_STATIC" = "1" ]
        then
            pkgfork subversion subversion-static
        fi
    menu_end
fi
```

Packages

Abstract

- Every package has a subdirectory in the package/ directory. E.g.: package/base/gcc
- Every package has a *.desc file with the metadata
- A semi-inteligent system autodetects how packages should be built
- Only some packages need dedicated build instructions in a *.conf file
- Patches (*.patch files in the package dir) are applied automatically
- So creating patches is extreamly easy with ROCK Linux

LINBIT Information Technologies GmbH

Repositories

- The packages are grouped in so-called repositories
- Every developer of developers-group has it's own repository
- E.g.: "base" for the core group
  or "clifford" for Clifford Wolf
- This is fully independent from the package categories
- There also is e.g. a "kde" repository for the kde base packages. That's to make switching maintainership for this packages easy and shound not be confused with the "desktop/kde" package category.

LINBIT Information Technologies GmbH

Example *.desc file


```
<package/base/qprofile/qprofile.desc>

[I] System-wide profiler for Linux systems

[T] QProfile is a system-wide profiler for Linux
[T] systems, capable of <...>

[A] John Levon <levon@movementarian.org>
[M] Clifford Wolf <clifford@clifford.at>

[C] base/system

[L] GPL
[S] Stable
[V] 0.8.1
[P] X -?--5--9 145.500

[D] 3487496302 qprofile-0.8.1.tar.gz http://dl.sourceforge.net
```

Example *.conf file


```
<package/base/qprofile/qprofile.conf>

install_pulpstone()      {
        cp -v $confdir/pulpstoner.pl        \
              $root/usr/bin/pulpstoner
        cp -v $confdir/pulpstonec.pl        \
              $root/usr/bin/pulpstonec
        cp -v $confdir/pulpstoned.sh        \
              $root/usr/bin/pulpstoned
        chmod +x $root/usr/bin/pulpstone[rcd]
}

hook_add  postmake  5 "install_pulpstone"
confopt="$confopt     --with-kernel-support"
```

Architectures

Abstract

- Every architecture has a subdirectory in the architecture/ directory.
- The architecture attributes are described by an archtest.out file.
- More complex architectures (with sub-architectures with e.g. multiple endianes), an archtest.sh can be used to hold the dynamic data.
- Various sub-systems may expect additional files.
  E.g. gcc is looking for a "gcc-options" file

Example architest.out file

<architecture/hppa/architest.out>

```
arch_sizeof_short=2
arch_sizeof_int=4
arch_sizeof_long=4
arch_sizeof_long_long=8
arch_bigendian=yes
```

Example archtest.sh file

<architecture/hppa/archtest.sh>

```
case "$ROCKCFG_HPPA_BITS" in
    32)
        arch_machine=hppa
        arch_target="hppa-unknown-linux-gnu"
        arch_sizeof_char_p=4 ;;
    64)
        arch_machine=hppa
        arch_target="hppa64-unknown-linux-gnu"
        arch_sizeof_char_p=8 ;;
esac
```

Example gcc-options file


```
<architecture/sparc/gcc-options>

if [ "$ROCKCFG_SPARC_OPT"       != "generic"    ] ; then
        var_append    GCC_WRAPPER_INSERT      \
                " " "-mcpu=$ROCKCFG_SPARC_OPT"

        tune=""
        case  "$ROCKCFG_SPARC_OPT"       in
                v7)  tune=cypress     ;;
                v8)  tune=supersparc      ;;
                v9)  tune=ultrasparc      ;;
        esac
        [ -n  "$tune"   ] && var_append   \
                GCC_WRAPPER_INSERT      " " "-mtune=$tune"

        if [ $ROCKCFG_SPARC_BITS      == 64  ] ; then
                var_append    GCC_WRAPPER_INSERT      " " \
                        "-Wa,-Av9a    -mno-app-regs"
        fi

fi
```

Targets

Abstract

- A ROCK Linux based distribution is called a "target"
- An installable CD-set usually consists of two targets:
  - A small boot-from-cd installer distribution
  - The distribution to be installed on the system
- Each target has it's own configuration (-cfg option)

Example config code

<target/crystal/preconfig.in>

```
CFGIEMP_TARGETLIST="$CFGIEMP_TARGETLIST crystal \
    Crystal_ROCK_(The_general_purpose_distriution)"
```

<target/crystal/config.in>

```
pkgfilter sed -e '/CORE/ ! s/^X/O/'
ROCKCFGSET_DO_REBUILD_STAGE=0
```

LINBIT Information Technologies GmbH

Example build code


```
<target/crystal/build.sh>

pkgloop

echo_header "Finishing build."

echo_status "Creating package database ..."
admdir="build/${ROCKCFG_ID}/var/adm"
create_package_db $admdir build/${ROCKCFG_ID}/ROCK/pkgs \
                  build/${ROCKCFG_ID}/ROCK/pkgs/packages.db

echo_status "Creating isofs.txt file .."
cat << EOT > build/${ROCKCFG_ID}/ROCK/isofs.txt
DISK1   $admdir/cache/                          ${ROCKCFG_SHORTID}/info/cache/
DISK1   $admdir/cksums/                         ${ROCKCFG_SHORTID}/info/cksums/
DISK1   $admdir/dependencies/                   ${ROCKCFG_SHORTID}/info/dependencies/
DISK1   $admdir/descs/                          ${ROCKCFG_SHORTID}/info/descs/
DISK1   $admdir/flists/                         ${ROCKCFG_SHORTID}/info/flists/
DISK1   $admdir/md5sums/                        ${ROCKCFG_SHORTID}/info/md5sums/
DISK1   $admdir/packages/                       ${ROCKCFG_SHORTID}/info/packages/
EVERY   build/${ROCKCFG_ID}/ROCK/pkgs/packages.db ${ROCKCFG_SHORTID}/pkgs/packages.db
SPLIT   build/${ROCKCFG_ID}/ROCK/pkgs/          ${ROCKCFG_SHORTID}/pkgs/
EOT
```

Wrappers

LINBIT Information Technologies GmbH

Filelist Wrapper

- Managing package file list by hand is waste of time
- ROCK Linux is using a so-called "flist wrapper" to monitor package build processes and create file lists automatically.
- The same mechanism is also used for auto-detecting build-time dependencies.
- The same mechanism is also used for auto-detecting build-time dependencies.

Command Wrapper (1/3)

- Patching Makefiles to hornour $CFLAGS or install files in different locations can be a pain in the neck.
- So ROCK Linux has the command wrapper infrastructure for dynamically rewrite commands executed by the build process.
- The common command wrapper is primarly used for mangling gcc options.
- Special wrappers exist for commands which are used to install files, such as cp and install.

Command Wrapper (2/3)

GCC Option Wrapper Examples:

<package/base/gcc/parse-config>

```
[...]
if [ $pkg != glibc23 -a $pkg != glibc22 -a \
     $pkg != grub -a $pkg != dietlibc ]
then
   if [ "$ROCKCFG_PKG_GCC_STACKPRO" = 1 ] ; then
      var_append GCC2_WRAPPER_INSERT \
         " " "-fstack-protector"
      var_append GCC3_WRAPPER_INSERT \
         " " "-fstack-protector"
   fi
fi
[...]
```

Command Wrapper (3/3)

Install Wrapper Examples:

<package/base/findutils/findutils.conf>

```
[...]
var_append    INSTALL_WRAPPER_FILTER        "|"  \
    "sed  -e  's,usr/bin/find,bin/find,'        \
          -e  's,usr/bin/xargs,bin/xargs,'"
[...]
```

<package/rene/xemacs-beta/xemacs-beta.conf>

```
[...]
var_append    INSTALL_WRAPPER_FILTER        "|"  \
    "sed  's,man1/xemacs\.1$,man1/xemacs-beta.1,'"
[...]
```

Some Other Features

LINBIT Information Technologies GmbH

Pseudo-Native Builds

- Doing native builds isn't usefull on some architectures (such as old Motorola 68000 systems).
- But cross-building is sometimes not possible (broken configure scripts and makefiles, etc).
- With "pseudo-native" builds it's possible to do a build which looks like a native build on an alien architecture.
- This is done by registering a small wrapper binary as handler for ELF binaries from the target architecture with the Linux kernel binfmt_misc hook.
- This wrapper either calls a local replacement runable on the host architecture or (in the remaining cases) runs the program on another machine of the target architecture.

LINBIT Information Technologies GmbH

Package Forks and Splits (1/2)

- One package description may dexcribe the build process for more than one package. E.g. there is only one gcc package in rock linux which currently handles the packages gcc2, gcc32, gcc33 and gcc34.
- This is possible due to a mechanism called "package forking".
- One package build process may produce more than one binary package. E.g. building the subversion package ends in the packages subversion, subversion:doc, subversion:dev, subversion:server and subversion:apache.
- This is possible due to a mechanism called "package splitting".

Package Forks and Splits (2/2)

Package Split Example Code:

<package/clifford/subversion/subversion.conf>

```
[...]
if pkginstalled apache; then
    apacheprefix=${ROCKCFG_PKG_APACHE_PREFIX:-opt/apache}
    [...]
    splitdesc_apache() {
        desc_1="Apache module for subversion"
    }
    splitreg 50 apache ${apacheprefix}.*mod_.*so
fi
[...]
```

LINBIT Information Technologies GmbH

Output Modules

- The output of the build process can be passed thru various output module.
- One of those output modules are used for the "normal" status output on the terminal.
- Some examples for other output modules are the modules for speech synthesis, html pages and mythtvosd.
- It's possible to use some ouput modules only for specific events.
  E.g. only using audio ouput using speech synthesis on errors.

Crystal ROCK (1/2)

- Most users don't really want to build their own distributions
- So ROCK Linux 3.0 will also contain a general purpose distribution called "Crystal ROCK".
- Crystal ROCK will be small (one binary CD for x86), 200-300 packages.
- It contains the most common desktop packages (such as KDE and all it's utilities) development tools (like gcc) and servers (like apache).

LINBIT Information Technologies GmbH

Crystal ROCK (2/2)

- Missing packages can be installed from binary package archives from the internet or built from source using the "rocket" tool.
- The "stone" program provides users with an easy-to-use config tool, but all the config files may also be edited directly and and primarly designed to be edited using a random ASCII editor.
- An alternative graphical installer is planed for the future.
- The Crystal ROCK target may be a good starting point for your own distributions.

URLs and References

URLs

- ROCK Linux Webpage
  http://www.rocklinux.org/
- ROCK Linux Portal
  http://www.rocklinux.net/
- Mailing Lists
  http://www.rocklinux.org/mail.html
- IRC Channel
  http://www.rocklinux.org/irc.html

Credits

- ROCK Linux Team
  http://www.rocklinux.org/gallery.html
- Clifford Wolf:
  http://www.clifford.at/
- LINBIT Information Technologies
  http://www.linbit.com/