lcrs-embedded

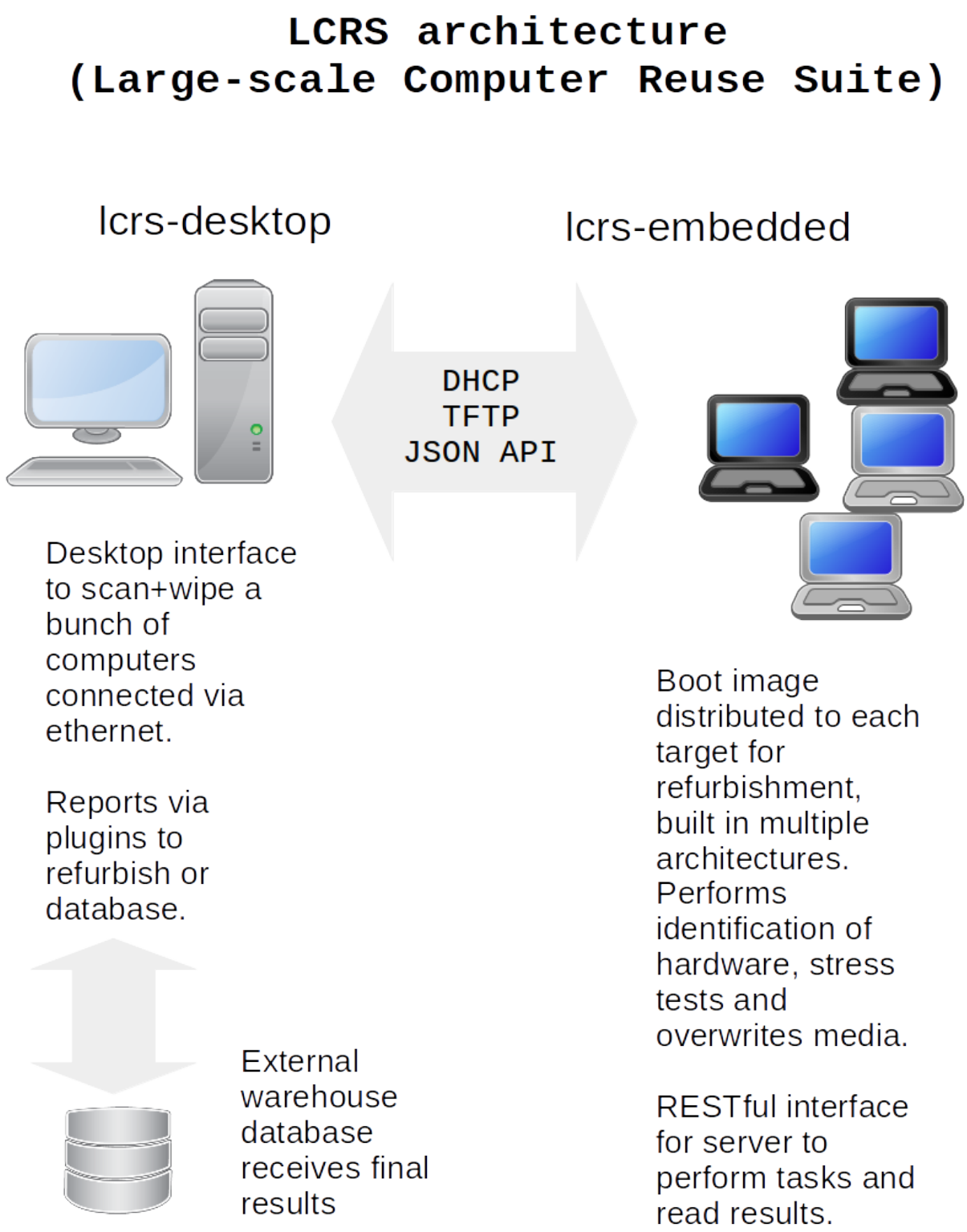
Release 1.0dev

Benjamin Bach

Jul 07, 2018

Contents

This is the documentation of the embedded part of Large-scale Computer Reuse Suite, a.k.a. LCRS.



CHAPTER 1

Large-scale Computer Reuse Suite (LCRS)

This is the embedded component of LCRS, a Linux kernel with an embedded server written in Python, using HTTP.

You can read more about LCRS in the main repo:

https://github.com/fairdk/lcrs

. . . and on our website:

https://lcrs.fairdanmark.dk Software for computer refurbishment workshops: Wipe disks, test hardware, store results in database.

- Free software: GNU General Public License v3
- Documentation: https://lcrs-embedded.readthedocs.io.

1.1 Contents

1.1.1 Installation

The embedded LCRS command line is available from within the built images that are booted by clients booted from the LCRS PXE server or from ISO images.

You may invoke it as a classic python library, but it serves no purpose on a regular system unless you want to wipe its hard drive.

To simulate usage, refer to the tests.

When installing within the embedded system, Buildroot is used. For installing it within this environment, a script is called invoking normal python setup procedure `pip install -e .` or `python setup.py install`, however it's also combined with installation of a relocatable virtualenv.

The concept is this:

```
mkdir -p buildroot/linked_buildroot/output/target/usr/bin
pip install . -t buildroot/linked_buildroot/output/target/usr/lib/python3.5/site-
↪packages/ --install-option="--install-scripts=buildroot/linked_buildroot/output/
↪target/usr/bin" --upgrade
```

(continues on next page)

(continued from previous page)

1.1.2 Developer guide

For an overview, refer to the Makefile and various entrypoints. This section contains guides to developing environments etc.

Contents:

Getting started

In order to build a kernel, you need to download Buildroot. You also need these dependencies:

```
sudo apt-get install build-essential ncurses-base ncurses-bin libncurses5-dev dialog␣
↪gcc-multilib g++ g++-multilib
```

Make sure to have the Buildroot manual at hand!

Copy the `.config` file from the Git repo into the location where you have unpacked Buildroot, then run the ncurses configuration program:

```
make nconfig
```

Buildroot configuration

In order to mess with the buildroot stuff and create new images containing the LCRS CLI, you need some build root skills.

The main entry point is the Makefile which will run scripts that prompt you for typical questions to get bootstrapped.

This is an documentation of the various choices made regarding the Buildroot configuration.

Make sure to have the Buildroot manual at hand!

Copying the `.config-dist` file should happen by running the Make target, if you have a buildroot environment at hand, run this to see the configuration:

```
make nconfig
```

Buildroot settings

- Root password: `unset`, meaning there's no password for the root user.
- Default DHCP device: `eth0`

Buildroot features

Before setting up the environment, consider that it takes quite a lot of storage space (~6 GB), so you might wanna put it on a different drive.

Furthermore, after building, you cannot relocate. You would have to rebuild. This is a well-known issue in Buildroot.

Toolchain:

Remember if you change the configuration of the toolchain, you need to rebuild everything with `make clean`.

- Wchar
- C++ support (because of smartmontools)

The following is compiled into the distributed Buildroot

System configuration:

- /dev management with mdev
- Network interface for DHCP: eth0

Packages

- bz2
- dt
- fio
- ramspeed
- stress
- cpio
- squashfs w/ gzip
- Linux binary firmware for all Ethernet
- dmidecode
- fan-ctrl
- hwdata
- kdb (keyboard tables)
- lm-sensors
- memtester
- pciutils (lspci)
- sdparm
- sg3utils w/ programs
- smartmontools
- sysstat
- wipe
- **python3**
  - All internal modules enabled
  - python-socketio

Libraries Hardware handling - lbusb

Networking -

Other - mcrypt

Text and terminal handling - libiconv - ncurses w/ wide-char (utf-8 handling) + ncurses programs - newt

Network applications:

- dhcpcd
- dhcpdump
- dropbear
- ethtool
- iputils
- macchanger
- netplug
- ntp

Shell utilities:

- None

System tools:

- cpuload
- htop
- keyutils

Text editors:

- nano

Maybe?

- ramspeed/smp
- stress-ng

Filesystem images

- ext2 root file system
- initial RAM filesystem linked into linux kernel
- iso image (isolinux)
- squashfs root

Bootloaders

- syslinux w/ isolinux + pxelinux

Test-driven development (TDD)

This program is written with TDD in mind.

The most sensitive parts of the suite of commands that are run are guided by a philosophy that all functions should guarantee their intended outcomes, but still have to rely on their respective system calls.

Warning: If something doesn't work, fail loudly. Throw exceptions, go mad.

Test philosophy

#. Define data structures, the main object to refer to is `lcrs_embedded.models.ScanResult`

API reference

lcrs_embedded

1.1.3 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/fairdk/lcrs-embedded/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

Large-scale Computer Reuse Suite (LCRS) could always use more documentation, whether as part of the official Large-scale Computer Reuse Suite (LCRS) docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/fairdk/lcrs-embedded/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *lcrs-embedded* for local development.

1. Fork the *lcrs-embedded* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/lcrs-embedded.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv lcrs-embedded
$ cd lcrs-embedded/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 lcrs-embedded tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/fairdk/lcrs-embedded/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_lcrs-embedded
```

1.1.4 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning

[Unreleased]

Added

- Example entry #1 @benjaoming

Changed

- Start using "changelog" over "change log" since it's the common usage.

Removed

- Section about "changelog" vs "CHANGELOG".