Roll your own Linux®, the easy way

LTIB Birds Of a Feather session

Stuart Hughes
Software Engineer
What is LTIB

• A tool to develop Linux® board support packages (BSPs)

• A tool to publish BSPs that are known to boot and run

• A tool to re-configure and rebuild published BSPs
  ▪ You can create your own based on existing ones
  ▪ You can re-publish (make your own ISO images)

• A tool to make all this (relatively) easy
What LTIB is not

• An SCM system
  ▪ Although some have been known to abuse it this way

• A Linux® distribution
  ▪ Some seem to think it is

• An application development environment
  ▪ Although it is useful if you need to add operating system components
Why do we need another target builder?

- Many fine projects, but no single project has all the required features:
  - Debian: Won't scale small enough
  - ELDK: Not easy to build from source
  - Buildroot: No package management, uClibc focus
  - OpenEmbedded: Too complex, scratchbox not available on all architectures
  - uClinux-dist: Monolithic download, no package management
Philosophy

• Open source (GPL)

• No proprietary internal data formats
  ▪ Uses kernel LKC for configuration, standard rpm spec files

• Simple command-line tool
  ▪ Text based so it's usable over low bandwidth links
  ▪ Can be driven by scripts and batched

• Common userspace package payload across all architectures

• All packages can be built from source (non-root user)
Philosophy (cont)

• Packages cross compiled with known good binary toolchains
  ▪ Sources available via srpms on GPP

• Content is kept separately from the build system
  ▪ Provided by packages pools (e.g. GPP)

• Target C library parts taken from the toolchain by default

• Don't gratuitously upgrade (bloat and spaghetti)

• Making a new target type should be easy
  ▪ The simplest could be just two text files
Features

• Runs on most popular Linux® distros (rpm or deb based)

• Supports multiple architectures (Power Architecture®, ARM®, Coldfire®)
  ▪ Can add new types if you have a cross toolchain and kernel

• Curses based configuration of kernel/packages/sysconfig/image

• Over 250 packages

• Auto package dependency resolution
Features (cont)

- Auto-conflict overlay (scaling)
- Auto-package dependency re-build/install trigger
  - e.g. coreutils removal will re-install busybox
- Can use your own custom toolchain or kernel
- Support for kernel/u-boot builds from directory or git trees
Features (cont)

• The kernel and busybox drop to their own config screens if required

• Support for uClibc or glibc

• Support for whole target image pre-configured node set (preconfigs)

• Support for pre-configured package sets (profiles)

• Interface headers/libraries/rpm database private per instance

• Spec files/cross compiling kept simple using 'spoofing'
Features (cont)

• Single package mode using prep/scbuild/scdeploy
  ▪ Modified sources are never automatically deleted

• Modified package sources can be captured using 'patchmerge'
  ▪ The corresponding spec file is also auto-updated with the new patch

• Semi-automated srpm import mode

• Shell mode to run at the command line in an LTIB environment

• NFS, RAMDISK and JFFS2 output supported
Features (cont)

• Incremental deploy to NFS root filesystem area

• Auto-builder support (--batch, --continue)

• Can list all available packages including details of licenses

• Release mode creates an ISO image including LTIB and packages
Basic Use

• Getting LTIB:
  • $ cvs -z3 -d:pserver:anonymous@cvs.savannah.nongnu.org:/sources/ltib co ltib

• Installing and building for the first time:
  • $ ./ltib

• Re-configuring:
  • $ ./ltib -m config

• Re-building:
  • $ ./ltib
Installing for the first time

$ cvs -z3 -d:pserver:anonymous@cvs.savannah.nongnu.org:/sources/ltib co ltib

....
U ltib/doc/index
U ltib/doc/wiki_style.css
$ cd ltib
$ ./ltib

Installing host support packages.

This only needs to be done once per host, but may take up to an hour to complete ...

If an error occurs, a log file with the full output may be found in: /home/seh/ltib/host_config.log
Initial configuration screen

Arrow keys navigate the menu. <Enter> selects submenus →. Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [ ] feature is excluded.

Platform choice (senTec COBRA5475 Coldfire/M68k (MMU)) →

- Load an Alternate Configuration File
- Save Configuration to an Alternate File

<Select> < Exit > < Help >
Selecting the target platform
Platform configuration screen

Choose the target C library type
- Target C library type (uclibc) →
- C library package (from toolchain only) →
- Toolchain component options →
  Choose your toolchain
  Toolchain (gcc-3.4.3-uclibc-0.9.28-nfp) →
() Enter any CFLAGS for gcc/g++
- Bootloader
  [ ] use cuImage boot capability
  [ ] See help: Build a boot loader

<Select>  < Exit >  < Help >
Package selection

Arrow keys navigate the menu. <Enter> selects submenus ——>. Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [ ] feature is excluded

[ ] dosfstools
[ ] dropbear ssh client/server
[*] non-blocking random device
[*] disable reverse host lookups
[*] disable X11 forwarding
[ ] use an insecure hackable RSA key
[ ] dtc
[ ] e2fsprogs
[ ] ed
[ ] ethtool

<Select>  < Exit >  < Help >
System configuration

Arrow keys navigate the menu. <Enter> selects submenus ——>. Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [ ] feature is excluded

(freescale) target hostname
[*] boot up with a tty and login
(:::respawn:/sbin/getty -L console 0 screen) Enter your init tab startup
() load these modules at boot
[ ] start devfsd
[*] start networking
   Network setup ——>
[*] set the system time at startup
(ntp.cs.strath.ac.uk) NTP server name/ip address
[*] start syslogd/klogd

<Select>  < Exit >  < Help >
Target image options

Choose your root filesystem image type

Target image: (jffs2) ——>
(64) jffs2 erase block size in KB (NEW)
[ ] read-only root filesystem
(512k) tmpfs size (NEW)
(/tmp /var) Place these dirs in writable RAM (NEW)
() rootfs target directory
[*] Keep temporary staging directory
[*] remove man pages etc from the target image
[*] remove the /boot directory

<Select>  < Exit >  < Help >
LTIB now builds the configuration chosen

Installing: tc-fsl-x86lnx-ppc-uclibc-nfp-3.4.3-1.i386.rpm

```
sudo /opt/ltib/usr/bin/rpm --dbpath /opt/ltib/var/lib/rpm -ivh --force --ignorearch /opt/freescale/pkgs/tc-fsl-x86lnx-ppc-uclibc-nfp-3.4.3-1.i386.rpm
```

Preparing... ################################################################### [100%]

1:tc-fsl-x86lnx-ppc-uclibc-3.4.3-1.i386 ################################################################### [100%]

Processing platform: A&MLtd Adder MPC875 PowerPC board

==============================================

using config/platform/qs875s/.config

Processing: fake-provides

==============================================

```
rpmbuild --dbpath /home/seh/ltib/rootfs/var/lib/rpm --target ppc --define
  '_unpackaged_files_terminate_build 0' --define '_target_cpu ppc' --define '__strip strip' --define
  '_topdir /home/seh/ltib/rpm' --define '_prefix /usr' --define '__tmppath /home/seh/ltib/tmp' --define
  '_mandir /usr/share/man' --define '_sysconfdir /etc' --define '_localstatedir /var' -bb --clean --rmsource
/home/seh/ltib/dist/lfs-5.1/fake-provides/fake-provides.spec
```
How it works

• Platform is selected from a list of directories in `config/platform/*`

• Platform is optionally re-configured using mconf
  ▪ Configuration saved in `config/platform/{target}/.config`

• ltb script reads configuration points to extract the package build list

• Build list is ordered by `config/userspace/pkg_map`

• Each package is built in order using a corresponding rpm spec file

• When all built, optionally a RAMDISK or JFFS2 image is built
How it works – building and installing a package

- Look up package .spec file
  - dist/lfs5.1/<pkg.spec>
  - Retrieve package files
    - base (.tgz or .bz2)
    - patches (.patch)
    - checksums (.md5)

- Install source tree
  - LPP
  - rpm/BUILD/<pkg name>

- Generate installable package (.rpm)
  - rpm/RPMS/<arch>/<pkg-arch.rpm>

- Compile package
- Install package (.rpm)
  - rootfs

GPP
How it works - spoofing

• Should not be needed, but some packages are not well behaved

• When ltib is building, gcc is an alias for the cross compiler

• Your per-project interface area is wired for you by spoofing
  ▪ You don't need to say: -l -L <rootfs>/usr/{include,lib}

• rpath-link is use to resolve indirect library dependencies

• The LTIB host support package pkg-config uses the <rootfs> prefix

• TOOLCHAIN_CFLAGS from ltib are always guaranteed to be injected
LTIB package pools

PPP/GPP/LPP data-flow

Internal to organisation

- Staging area
  - Review
    - PPP
    - CGI
  - LPP
  - Browser

The Internet

- CGI
  - GPP

End user Linux PC

- LPP
  - Browser
More advanced command line options

• Modes:
  • Single package: `-m prep/scbuild/scbuild/scinstall/scdeploy/patchmerge`
  • Erase packages: `-m clean`
  • Start again: `-m distclean`
  • List packages: `-m listpkgs`
  • Make an ISO: `-m release`
  • Configure only: `-m config`
  • Shell mode: `-m shell`
More advanced command line options (cont)

- Options:
  - One package only: `--pkg <pkg>`
  - Configure and build: `--configure`
  - Whole configuration: `--preconfig <filename>`
  - Use these packages: `--profile <filename>`
  - Batch mode: `--batch`
  - Disable dependency: `--nodeps`
  - Conflict check on: `--conflicts`
  - Create srpms: `--keepsrpms`
  - Verbose output: `--verbose`
  - Dry run: `--dry-run`
  - Continue on error: `--continue`
More advanced command line options (cont)

- Options (cont):
  - Output version: `--version`
  - Download only: `--dlonly`
  - Download test: `--dltest`
  - Leave built sources: `--leavesrc`
  - Host packages: `--hostcf`
  - Help screen: `--help`
Resources

• LTIB home page:
  ▪ http://www.bitshrine.org/

• LTIB project, including CVS (hosted by Savannah)
  ▪ http://savannah.nongnu.org/projects/ltib

• LTIB mailing list (hosted by Savannah)
  ▪ http://lists.nongnu.org/mailman/listinfo/ltib

• Freescale BSP ISO releases (free to download and use)
  ▪ http://www.freescale.com/webapp/sps/site/overview.jsp?code=CW_BSP&srch=1
Demo and Questions

Demo
  • Time/hardware permitting

Questions?
  • Ask me now
  • Send email to: stuarth at freescale dot com

Thank you for attending!