Bootstrapping the Debian and Ubuntu ARM64 Ports

Wookey

Linaro

17th November 2013

Who am I

- Free Software developer since 1990
- Unix sysadmin since 1996
- Arm Linux developer since 1999
- Debian developer since 2000
- Ubuntu development since 2010

Some things I had something to do with:
Survex, PsiLinux, ArmLinux book, Emdebian, bootfloppies, Therion, apt-cross, dpkg-cross, Debian cross-toolchains, OpenEmbedded, Netbook Project, LART, YAFFS, Balloonboard, xdeb, multiarch, sbuild

- Currently an ARM secondee to Linaro

Linaro

Outline

Linaro

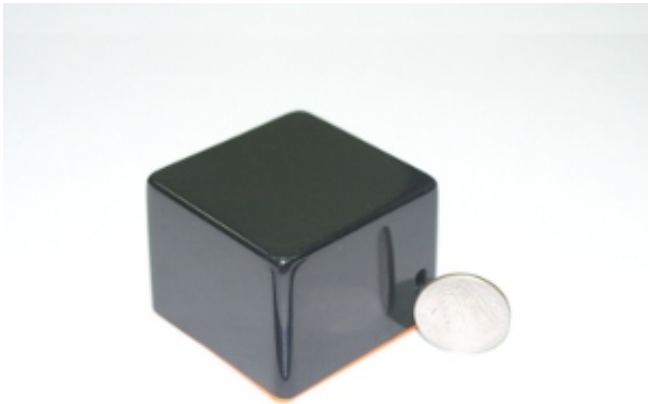
Outline

Linaro

ARM desktops and servers

Acorn Risc PC (1994)

Rebel Netwinder (2000)



Solidrun Cubox (2012)

Dell/Calxeda server (2012)

Linaro

ARM laptops

Psion Netbook Pro (2003)

Toshiba AC100 (2010)

Genesi Smartbook (2010)

Samsung Chromebook (2012)

Linaro

Debian ports

Name	Bits	ABI	ISA	Released
arm	32	OABI	v3	2000:Potato (discontinued 2011)
armeb	32	OABI	v3	2006:unofficial big endian
armel	32	EABI	v4t/v5	2009:Lenny, Ubuntu 9.05
armhf	32	EABI	v7	2012:Wheezy, Ubuntu 12.04
arm64	64	v8	v8	2013:Jessie?, Ubuntu 13.10 (Saucy)

Linaro

Nomenclature (architectures)

Simple version

arm64, aarch64, ARMv8 are all the same thing

More details:

arm64	Debian and Ubuntu architecture name
aarch64	ARM 64-bit execution mode
aarch64-linux-gnu	GNU triplet name
ARMv8	ARM CPU architecture name
A64	64-bit instruction set
A32	32-bit ARMv8 instruction set
aarch32	ARM 32-bit execution mode

Linaro

Nomenclature (building)

`Build` : Machine/architecture you are building on

`Host` : Machine/architecture package is being built for

`Target` : Machine/architecture a compiler generates code for

Bootstrapping

22 Debian ports in 20 years

i386, 68000, Alpha, Sparc, PowerPC, ARM, IA64, PA-RISC, MIPS (big endian), MIPS (little endian), S/390, AMD64, FreeBSD-i386, FreeBSD-amd64, armel, armhf, sh4, s390x, PowerPC64, Hurd-i386, Mips64el

Linaro

Bootstrapping

22 Debian ports in 20 years

i386, 68000, Alpha, Sparc, PowerPC, ARM, IA64, PA-RISC, MIPS (big endian), MIPS (little endian), S/390, AMD64, FreeBSD-i386, FreeBSD-amd64, armel, armhf, sh4, s390x, PowerPC64, Hurd-i386, Mips64el

Bootstrapping is normal, not exceptional
A 'Universal OS' should be able to bootstrap itself

Linaro

Outline

Linaro

The Bootstrap Problem

- Build-dependency loops

The Bootstrap Problem

- Build-dependency loops

A more detailed look

The Bootstrap Problem

- Build-dependency loops
- Natively built
- Maximally configured
- Much worse for binary distros than source-based
- Lack of flexibility is in packaging

Linaro

Outline

Linaro

Bootstrap solutions

Traditionally

- Cheat and use something else
- Bodgery and Hackery
- No hardware yet - models are really slow

'Universal OS' solution

- Cross Build at least initial chroot
- Linearise build order by reducing dependencies
- Switch to native building when you have *'enough'*

Linaro

Build Profiles

debian/control

```
1  Build-Depends:  debhelper,..., libsql-dev
   Build-Depends-stage1:  debhelper,...

2  Build-Depends:  debhelper,..., libsql-dev <!stage1>

3  Build-Depends:  debhelper,..., libsql-dev [profile.!stage1]
```

https://wiki.debian.org/BuildProfileSpec

debian/rules

```
ifeq ($(DEB_BUILD_PROFILE),stage1)
DH_OPTIONS += -Nlibdb5.1-sql
  CONFIGURE_SWITCHES += --disable sql
else
  CONFIGURE_SWITCHES += --enable-sql
fi
```

Linaro

Multiarch

- Install libraries side-by side: i386/amd64, arm/arm64, amd64/arm64
  - /usr/lib/libfoo (amd64)→/usr/lib/x86_64-linux-gnu/libfoo
  - /usr/lib/libfoo (armel)→/usr/lib/arm-linux-gnueabi/libfoo
  - /usr/lib/libfoo (arm64)→/usr/lib/aarch64-linux-gnu/libfoo
- Packages arch-qualified: libfoo:arm64, wine:i386
- Canonical file locations: Runtime is the same as build-time.
- Run foreign binaries in-place (natively or with qemu)
- 32/64 special casing goes away (/lib64, /emul/ia32-linux)
- Build/host version lockstep

Usage example

```
dpkg --add-architecture i386
apt-get install skype:i386
```

Linaro

Multiarch dependencies

Packages are given an extra field Multi-Arch

- same *(libraries)*
  can be co-installed and can only satisfy deps within the arch
- foreign *(tools)*
  can not be co-installed can satisfy deps for any arch
- allowed *(both)*
  can be either. Depending packages specify which is wanted

dpkg has support for reference-counting of (doc-) files from co-installable packages that overlap

Modifed by

:any	treat M-A:allowed package as M-A:foreign (e.g. perl:any)
:native	install build-arch version (e.g. libnih-dbus-dev:native)

Cross-dependencies

Example: slang2

```
Build-Depends: debhelper (>= 9), autoconf, autotools-dev,
       chrpath, docbook-to-man, dpkg-dev (>= 1.16.1~),
       libncurses-dev,
       libpcre3-dev,
       libpng-dev,
       zlib1g-dev
```

- Build-arch: debhelper, autoconf, autotools-dev, chrpath, docbook-to-man, dpkg-dev
- Host-arch: libncurses-dev, libpcre3-dev, libpng-dev, zlib1g-dev

```
apt-get install debhelper autoconf autotools-dev chrpath docbook-to-man
dpkg-dev libncurses-dev:arm64 libpcre3-dev:arm64 libpng-dev:arm64
zlib1g-dev:arm64
```

Outline

Linaro

ARM internal Bootstrap (2011)

- Ubuntu Maverick
- Using xdeb, with staging support
- Equivs to fake toolchain dependencies
- Manual build order

Linaro

So it's all done already?

ARM is an IP Company

So it's all done already?

ARM is an IP Company

Now I can be rude about ARM legal

So it's all done already?

ARM is an IP Company

Now I can be rude about ARM legal

- Paranoid about patent grants in FLOSS licences
- No cross-fixes, bootstrapping or arm64 support upstreamed
- Engineers annoyed

Linaro

So it's all done already?

ARM is an IP Company

Now I can be rude about ARM legal

- Paranoid about patent grants in FLOSS licences
- No cross-fixes, bootstrapping or arm64 support upstreamed
- Engineers annoyed
- All has to be done again

Linaro

Valuable IP - avert your eyes:

```
+export DEB_BUILD_GNU_TYPE ?= $(shell dpkg-architecture -qDEB_BUILD_GNU_TYPE)
+
+ifeq ($(DEB_BUILD_GNU_TYPE), $(DEB_HOST_GNU_TYPE))
+  confflags += --build $(DEB_HOST_GNU_TYPE)
+  CROSS=""
+else
+  confflags += --build $(DEB_BUILD_GNU_TYPE) --host $(DEB_HOST_GNU_TYPE)
+  CROSS=$(DEB_HOST_GNU_TYPE)-
+endif
```

On the one hand great early community engagement

On the other complete failure to give back

Illustrates Linaro/corporate culture clash

Outline

Linaro

Debian/Ubuntu Bootstrap Overview

Overview

- Initially Quantal, then Raring -based
  (and Debian Wheezy/Experimental)
- All done in public from start - upstreaming as we go along
- Multiarch building and cross-dependencies
- Standard tools: sbuild, reprepro, apt, dpkg, dpkg-cross
- Modified dpkg, apt, sbuild for build-profile support
- cross-build-essential: toolchain, libc:arm64, <triplet>-pkg-config
- no qemu available

Linaro

Debian/Ubuntu Bootstrap Process

1. Prepare repository
2. Add new arch support to dpkg-architecture
3. Set up build chroot
4. Toolchain bootstrap
5. Fix support packages: dpkg-cross, cross-build-essential, autoconf
6. Build stuff...

Linaro

How much 'stuff' do we need?

Binary(source) packages needed

	Debian Sid	Ubuntu Saucy Core
Base system	116	128
+ `build-essential`	128	140
Number of source packages	65	75
Sources including build-deps	119	
Number of binary packages	503	

Linaro

Set up a chroot

http://wiki.linaro.org/Platform/DevPlatform/CrossCompile/arm64bootstrap

Create chroot

```
apt-get install sbuild
sudo sbuild-createchroot
  --make-sbuild-tarball=/srv/chroots/raring-cross-arm64.tgz raring
  /srv/chroots/raring http://archive.ubuntu.com/ubuntu/
```

Build flags

```
STRIP CFLAGS -fstack-protector
```

Apt preferences

```
Package:  *
Pin:  release n=raring-bootstrap
Pin-Priority:  1001
```

Building Packages

Getting Build-Deps and building is simple

Manually

```
apt-get install crossbuild-essential-arm64
apt-get build-dep -aarm64 acl
apt-get source acl; cd acl-2.2.51
dpkg-buildpackage -aarm64
```

Better

```
CONFIG_SITE=/etc/dpkg-cross/cross-config.arm64
     DEB_BUILD_OPTIONS=nocheck dpkg-buildpackage -aarm64
```

Using sbuild

```
sbuild -c raring-bootstrap -d raring
  --host=arm64 acl_2.2.51
```

Profiled Package Build

Manually

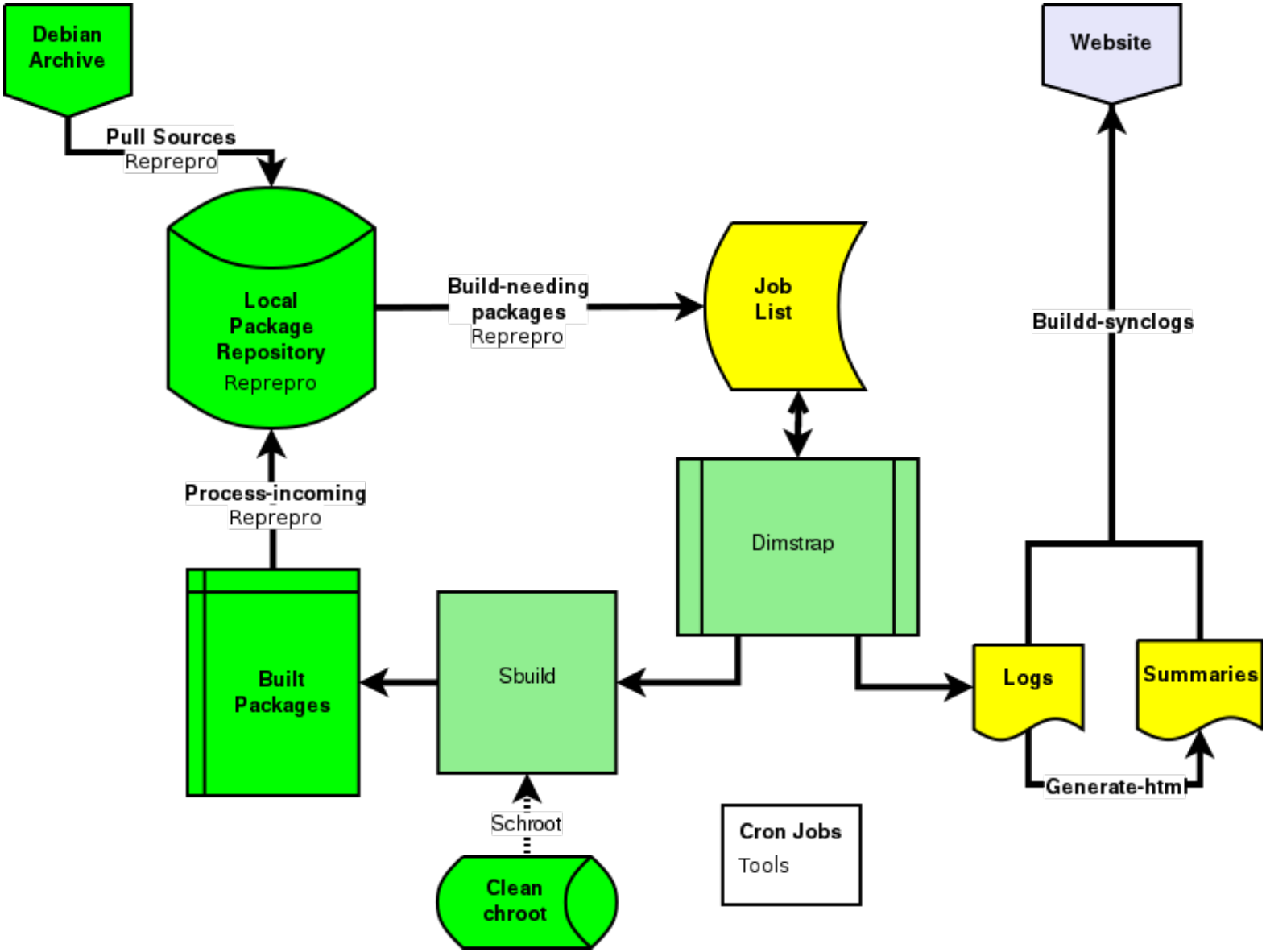
```
apt-get install crossbuild-essential-arm64
apt-get -o APT::Build-Profile=stage1 build-dep -aarm64 acl
apt-get source acl; cd acl-2.2.51
DEB_BUILD_PROFILE=stage1 dpkg-buildpackage -aarm64
```

Using sbuild

```
sbuild -c raring-bootstrap --profile=stage1 -d raring
  --host=arm64 acl_2.2.51
```

- update, build, sign, upload, process loop scripted with dimstrap

Cross Build Daemon



Dependency analysis

Dependency analysis

```
dose-debbuildcheck --deb-native-arch=amd64
--deb-foreign-archs=arm64 --deb-host-arch=arm64 <packages
files> <source file> -f -e -s --checkonly <package>
```

Output

```
package: src:dpkg
version: 1.16.7ubuntu3profile1
architecture: any,all
essential: false
unsat-dependency: arm64:liblzma-dev
```

Linaro

New tools

http://bootstrap.debian.net/

Daily analysis of 'bootstrapability'
Runs botch - 'Boot Ordering Tool'
Proposed to link this from the BTS: #728298

https://gitorious.org/debian-bootstrap/gsoc2013

Tool to automate bootstrapping
Uses pre-calculated botch ordering
Sets up repos, Builds from snapshot
Supports native and cross bootstrapping

Linaro

Modified Packages/Build-deps

Build-deps skipped with profiles - 10 packages

eglibc:	libselinux, libaudit
libselinux:	gem2deb, swig, python-all-dev
libsemanage:	gem2deb, swig, python-all-dev
cracklib2:	python-all-dev, python3-all-dev, python-setuptools, python3-setuptools
dbus:	libdbus-glib-1-dev, libglib2.0-dev, libsystemd-daemon-dev, libsystemd-login-dev, python, python-dbus, python-gobject
db:	gcj-native-helper, javahelper, default-jdk
glib2.0:	python-dbus, python-gi, dbus, dbus-x11
gnupg:	libldap2-dev
linux:	binutils-dev, libaudit-dev, libunwind8-dev, libnewt-dev, libelf0-dev, libdw-dev
udev:	libgirepository1.0-dev, gir1.2-glib-2.0

Outline

Debconf Activity

- Asked for debian-ports space at Debconf13:

  Machine is too full

  - Port instance appeared $<$ 1 week later
- Embedded interpreters multiarch discussion
  `https://wiki.debian.org/Multiarch/InterpreterProposal`
  Python multiarched in Ubuntu. Debian?
  Multiarch perl in `git://anonscm.debian.org/perl/perl.git`

Linaro

Debian Bootstrap

Method

- Native build Debian sources in Saucy chroot
- Nobbled dpkg origin and lsb_release info
- Clean Saucy tarball chroot + debianise script
- All deps available - take care with libs
- pin bootstrap repo as preferred

No hardware

- 80-core, 128G Xeon box in Huawei lab - no root access
- Model very slow and annoying (X, network tap)
- qemu-arm64 released - Way better!

Linaro

Qemu-arm64

- Developed by SuSE
- Userspace only
- `https://github.com/susematz/qemu/tree/aarch64-1.6` qemu-arm64 branch
- `https://wiki.debian.org/Arm64Qemu`

/usr/share/binfmts/qemu-arm64

package qemu-user-static
interpreter /usr/bin/qemu-arm64-static
flags: OC
offset 0
magic \x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\xb7
mask \xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff

Debian Bootstrap Status

Issues

- Undefined instructions - (tests)
- Java doesn't install or run
- Cyclic dependencies
- cc1 has different magic
- perl 5.18 vs 5.16
- extra /etc/lsb-release file

Status

- Arch all packages are easy
- 119 source Packages built

Linaro

Debian Bootstrap Status

Issues

- Undefined instructions - (tests)
- Java doesn't install or run
- Cyclic dependencies
- cc1 has different magic
- perl 5.18 vs 5.16
- extra /etc/lsb-release file

Status

- Arch all packages are easy
- 119 source Packages built

Rootfs debootstrapped 6pm yesterday

Linaro

Getting involved

Help is welcome - much easier with qemu

Just trying arm64

- `https://wiki.debian.org/Arm64Port#Pre-built_Rootfs`

Resources

- `http://wiki.debian.org/Arm64Port`
- `http://people.debian.org/~wookey/bootstrap.html`

Linaro

Thanks

Wookey
wookey@wookware.org
http://wookware.org

http://wiki.debian.org/Arm64Port