Multiarch - what is it and why should I care?

Wookey

Linaro and Debian

25th April 2014
openSUSE Conference
Dubrovnik, Croatia

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
- Currently an ARM secondee to Linaro
- Here for some cross-distro de-siloing

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, Debian armel and arm64 ports, Balloonboard, xdeb, multiarch, sbuild, build profiles

Linaro

Mulitarch

- Technical:
  - What is it?
  - What does it do?
  - How does it work?
- Social:
  - Is it important?
  - What's it like making big changes like this?
  - What did we learn from the excercise?

Linaro

Nomenclature

`Build` : Machine/architecture you are building on

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

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

Linaro

Outline

Linaro

What is Multiarch?

Multiarch is a general mechanism for installing libraries of more than one architecture (ABI) on a system

- more general than /lib,/lib64
- simple enough in principle . . .

Linaro

Multiarch is very simple

- Put libraries into architecture-specific paths
  - /usr/lib/libfoo (i386)→/usr/lib/i386-linux-gnu/libfoo
  - /usr/lib/libfoo (armhf)→/usr/lib/arm-linux-gnueabihf/libfoo
  - /usr/lib/libfoo (arm64)→/usr/lib/aarch64-linux-gnu/libfoo
- Change the default linux loader path.

The fundamental thing is that libraries have a canonical path

- Native and non-native locations are the same
- Cross -build and -runtime locations are the same
- Scrap 32-bit emulation dir (/emul/ia32-linux)
- Largely removes need for sysroots, and multilib

Linaro

Multiarch Genesis

Debian never used /lib64 for amd64 (or sparc or mips)

Native libs should be in /lib

alpha, ia64, s390	/lib
amd64, sparc64, ppc64	/lib64
x32	/libx32

Multiarch Genesis

Debian never used /lib64 for amd64 (or sparc or mips)
Native libs should be in /lib

alpha, ia64, s390	/lib
amd64, sparc64, ppc64	/lib64
x32	/libx32

32-bit compatibility libs built as foo-i386 packages, put in /lib32
Many collected into massive ia32-libs package.

Linaro

Multiarch Genesis

Debian never used /lib64 for amd64 (or sparc or mips)
Native libs should be in /lib

alpha, ia64, s390	/lib
amd64, sparc64, ppc64	/lib64
x32	/libx32

32-bit compatibility libs built as foo-i386 packages, put in /lib32
Many collected into massive ia32-libs package.

> *ia32-libs [is now] the biggest source package in Debian. [. . .]*
>
> *Tollef Fog Heen*
> *2005-07-10*

340MB, containing 128 libraries
ia32-libs was always intended as a temporary solution
Unfortunately the proper replacement took more than 6 years to arrive.

Outline

Things Multiarch does

- Arbitrary mix of 64 and 32-bit apps
- Cross-grading from one architecture to another
  - (armel→armhf, i386→amd64, armhf→arm64)
- Cheap emulated environments - emulate only the parts you need to
- Cross-compilation is no longer special - you get it for free
- Removes packaging complexity
- Simple support for binary-only software
  - (flash-plugin, skype, java-plugin, steam . . .)

Linaro

Things Multiarch doesn't do

- Install more than one arch of binaries/tools in /bin
  - So you still only get one arch of each app at a time
- Specify ABI-compatible capabilities (SSE/NEON/ALTIVEC/MMX, armv5/armv7, i486/i686)
  - ABI is calling convention, not instruction set
  - Complimentary to multilib for optimisations

Linaro

Things Multiarch allows

- Partial architectures
- Cross-dependencies (e.g for cross-compilers)
- Co-installable compilers (e.g. i386 on amd64 for ghc)
- Automatic port Bootstrapping (with build profiles)
- Other increasingly crazy stuff. . .

Linaro

Outline

Linaro

How it works - paths

GNU triplets are used for architecture paths, with some adjustment for historical cruft

Debian arch	GNU triplet	Multiarch library path
amd64	x86_64-linux-gnu	/usr/lib/x86_64-linux-gnu
i386	i486-linux-gnu	/usr/lib/i386-linux-gnu
i386	i586-linux-gnu	/usr/lib/i386-linux-gnu
armel	arm-linux-gnueabi	/usr/lib/arm-linux-gnueabi
ppc64	powerpc64-linux-gnu	/usr/lib/powerpc64-linux-gnu

- dpkg-architecture -qDEB_HOST_MULTIARCH returns pathname
- an equivalent lsb mechanism is needed for upstream and non-debian distros

Linaro

How it works - co-installability

Multi-arch-ready 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 reference-counts arch-independent files from co-installable packages that overlap

Linaro

Multiarch in use

dpkg (low-level package-manager)

dpkg controls 'enabled' architectures. Native arch is special, others are all 'foreign'.

`dpkg --add-architecture i386`
Enables a new (foreign) architecture

`dpkg --remove-architecture i386`
Takes it away again. It won't let you do that until there are no packages of that arch installed on the system.

Multiarch in use

apt (high-level package manager)

apt source entries get an (optional) arch field

```
deb [arch=amd64,i386] http://archive.ubuntu.com/ubuntu saucy main
deb [arch=armel] http://ports.ubuntu.com/ saucy main
deb-src http://ports.ubuntu.com/ saucy main
```

apt has two important config options:

`APT::Architecture` (existing option)
: Arch to use when fetching and parsing package lists

`APT::Architectures` (new option)
: All supported arches (native + foreign)

Multiarch in use

To use it, just specify required foreign arch on end of package name.

```
apt-get update
apt-get install libattr1-dev:i386
```

dpkg now shows these packages installed:

```
libattr1                    install
libattr1:i386               install
libattr1-dev                install
libattr1-dev:i386           install
libc6                       install
libc6:i386                  install
libc6-dev                   install
libc6-dev:i386              install
```

Multiarch in use - proprietary binaries

(Sorry this slide is all in debian-ese)

```
dpkg --add-architecture i386
apt-get update
wget -O skype-install.deb http://www.skype.com/go/getskype-linux-del
dpkg -i skype-install.deb
apt-get -f install
```

If the vendor has a repo it's even simpler:

```
dpkg --add-architecture i386
echo <vendor repo> > /etc/apt/sources.list.d/<vendor>.list
apt-get update
apt-get install <package>:i386
```

Multiarch in use - Crossgrading

(More debian-ese)

```
dpkg --add-architecture amd64
apt-get update
apt-get install linux-image-amd64:amd64
reboot
apt-get clean
apt-get --download-only install dpkg:amd64 apt:amd64
dpkg --install /var/cache/apt/archives/*_amd64.deb
```

Linaro

Crossbuilding

Cross building has 3 major issues

- Installing build dependencies: native tools, cross libs/headers
- Finding/linking libraries
- Running build-time tools

Multiarch helps with all of them.

Linaro

Cross-dependencies

Example: acl

```
Build-Depends: debhelper (>= 9), autoconf, autotools-dev
       dpkg-dev (>= 1.16.1~), libattr1-dev
```

- Build-arch: debhelper, autoconf, autotools-dev, dpkg-dev
- Host-arch: libattr1-dev

```
apt-get -aarm64 build-dep acl

The following NEW packages will be installed:
  autoconf automake autotools-dev bsdmainutils debhelper dpkg-dev
  gcc-4.8-base:arm64 gettext gettext-base groff-base
  html2text intltool-debian libattr1:arm64 libattr1-dev:arm64
  libc6:arm64 libc6-dev:arm64 libcroco3 libgcc1:arm64
  libgettextpo0 libpipeline1 libtool libunistring0 libxml2
  linux-libc-dev:arm64 m4 man-db po-debconf
```

Crossbuilding - library paths

Runtime is the same as build-time.

Old system (classic/dpkg-cross)

build-time library path:	/usr/arm-linux-gnueabi/lib/libfoo
runtime library path:	/usr/lib/libfoo

Sysroot

build-time library path:	/$sysroot/usr/lib/libfoo
runtime library path:	/usr/lib/libfoo

Multiarch

build-time library path:	/usr/lib/arm-linux-gnueabi/libfoo
runtime library path:	/usr/lib/arm-linux-gnueabi/libfoo

Much harder for libtool to screw it up :-)

Crossbuilding - build-time tools

- Can just run them with qemu
- Tool dependencies easily specified
- All fractions of system-emulation easily accomodated

Building Packages

Getting Build-Deps and building is simple

Manually

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

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
```

Headers and Dev packages

Dev packages need converting in order to be able to install both native and foreign versions.

- Move include files which differ between arches into /usr/include/<triplet>/
- Moving out all binaries. (usually foo-config - move to pkg-config)
- Get rid of .la files if possible

Changes required in implementation

Obviously many packages are affected (all libs, most -dev, some tools)

Also various core tools (anything that knows about library paths)
Had to be fixed before other uploads. (Not all of these were expected):

- libc (loader)
- dpkg, apt
- compilers (default library and header paths)
- make (foo: -lbar syntax)
- pkg-config
- pmake
- cmake
- debhelper
- lintian
- libffi
- openjdk (lib-jna)
- dpkg-cross
- klibc

Linaro

Future possibilities

Coinstallable binaries

- Would be useful
- Deliberately left out of initial implementation
- Needs a spec defining

Non-posix architectures (mingw64-amd64, mingw64-i386)

- Remove 277 mingw-w64 OpenSuse packages
- normal cross-toolchains
- Automatic build-dependencies
- libc headers is the trickiest bit.
- Stephen Kitt is working on it

Linaro

Remaining issues - loader clashes

- ld.so clashes prevent some architecture combinations
  - hppa/m68k/ppc32/s390 (/lib/ld.so.1)
  - i386/sh/alpha/sparc32 (/lib/ld-linux.so.2)

- Fortunately not ones too many people care about
  `https://sourceware.org/glibc/wiki/ABIList` is definitive

Linaro

Remaining issues - Embedded interpreters

The multiarch model assumes c-style library dependencies

- liba:arch → libb:arch → libc:arch

Interpreted languages break this assumption

- modulea:arch → moduleb:whatever → modulec:arch

Linaro

Outline

Multiarch timeline

- 2004 BOF at Debconf 4
- 2005 Talk at Debconf 5
- 2006 FOSDEM multiarch meeting
- 2008.06 Dpkg multiarch patches uploaded
- 2009.05 apt and dpkg maintainers agree on a package management spec at UDS in Barcelona. Scope restricted to libs.
- 2010.08 Tuple proposal for ABI names drafted (Debconf 10)
- 2011.02 dpkg multiarch implementation (sponsored by Linaro) lands in Ubuntu, proposed to LSB. Tuples revised to GNU Triplets.
- 2011.04: Ubuntu 11.04 released with 83 libraries multiarched, and 14 in a ppa: enough to cross-install flash plugin
- 2012.04: Multiarch Dpkg uploaded to Debian Unstable
- 2012-2014: Lots of packages converted.

Linaro

Multiarch now

Ubuntu precise (April 2012)
110 out of 112 (source) libs (main) (98%)

Ubuntu raring (April 2013)
372 source out of 399 source libs (main) (93%)
472 source out of 2941 source packages (main)

Debian Stable (Feb 2014)
684 source out of 1303 source libs (52%)
831 source out of 17161 source packages

Debian Unstable (now)
1112 source out of 1246 source libs (89%)
1385 source out of 19402 source packages

Linaro

Outline

Things we learned along the way

This is a classic example of a significant distro/ecosystem-wide change. These things are (*very*) hard to get done.

- Use written specs to record shared understanding

Linaro

Things we learned along the way

This is a classic example of a significant distro/ecosystem-wide change. These things are (*very*) hard to get done.

- Use written specs to record shared understanding

Linaro

Things we learned along the way

This is a classic example of a significant distro/ecosystem-wide change. These things are (*very*) hard to get done.

- Use written specs to record shared understanding
- Limit scope to something manageable (resist creep!)
- Get affected parties involved early
- Split your work into bite-sized deliverables
- Make it clear how people can help: http://wiki.debian.org/Multiarch/Implementation
- Design to avoid flag days

Linaro

But it's only Debian

- Work done in distro because that's practical
- Wider adoption always assumed

Linaro

But it's only Debian

- Work done in distro because that's practical
- Wider adoption always assumed
- In practice enthusiasm varied
- We thought we had everyone relevant involved - we were wrong

Linaro

But it's only Debian

- Work done in distro because that's practical
- Wider adoption always assumed
- In practice enthusiasm varied
- We thought we had everyone relevant involved - we were wrong
- ARM had to back out multiarch paths from aarch64 to get upstream

Linaro

Why does Multiarch matter?

Significant development of UNIX/FHS/LSB

Debian ecosystem has done the hard work and shown that it works
Unifies many other things which solve a piece of the problem
Is this useful (enough) beyond Debian and derivatives?

Discuss. . .

Linaro

Multiarch - What is it and why should I care?

Thanks!

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

about the slides:
available at http://wookware.org/talks/

Linaro

Useful URLs

- http://wiki.debian.org/Multiarch

  Index to specs, instructions for packagers, historical docs

- https://wiki.ubuntu.com/MultiarchSpec

  The main multiarch spec

- ttp://wiki.debian.org/Multiarch/Implementation

  HOWTO for packagers

Linaro