Multiarch in Debian: six months (or six years) on

Steve Langasek
vorlon@debian.org

26 July 2011
ia32-libs [is now] the biggest source package in Debian.
-Tollef Fog Heen
ia32-libs [is now] the biggest source package in Debian.

-Tollef Fog Heen, 2005-07-10
The problem has been known for a long time; the solution is six years late
Multiarch: recent history

- May 2009: apt and dpkg maintainers agree on a package management spec at UDS in Barcelona
- Aug 2010: BoF at DebConf in New York proposed new directory names
- Feb 2011: dpkg multiarch implementation (sponsored by Linaro) lands in Ubuntu
- Mar 2011: new directory names scrapped, but finalized (and normalized) GNU triplets adopted
- Apr 2011: Ubuntu 11.04 released with 83 libraries multiarched, +14 in a ppa: enough to cross-install flash plugin
- July 2011: 135 libraries multiarched in Debian
Multiarch: recent history

- May 2009: apt and dpkg maintainers agree on a package management spec at UDS in Barcelona
- Aug 2010: BoF at DebConf in New York proposed new directory names
- Feb 2011: dpkg multiarch implementation (sponsored by Linaro) lands in Ubuntu
- Mar 2011: new directory names scrapped, but finalized (and normalized) GNU triplets adopted
- Apr 2011: Ubuntu 11.04 released with 83 libraries multiarched, +14 in a ppa: enough to cross-install flash plugin
- July 2011: 135 libraries multiarched in Debian
- July 28 2011: ??
Multiarch: key lessons

Multiarch is an example of how to - and how not to - make large changes to Debian that require coordination between many different maintainers.

- Use written specs to record **shared** understanding (not just for Debian Policy)
Multiarch: key lessons

Multiarch is an example of how to - and how not to - make large changes to Debian that require coordination between many different maintainers.

- Use written specs to record shared understanding (not just for Debian Policy)
- Split your work into bite-sized deliverables (avoid needing to do years of work before anyone else can see the results!)
- Make it clear how people can help
Multiarch: key lessons

Multiarch is an example of how to - and how not to - make large changes to Debian that require coordination between many different maintainers.

▶ Use written specs to record **shared** understanding (not just for Debian Policy)
▶ Split your work into bite-sized deliverables (avoid needing to do years of work before anyone else can see the results!)
▶ Make it clear how people can help
  http://wiki.debian.org/Multiarch/Implementation
▶ There is nothing so permanent as a temporary solution.
Multiarch: so what?

What does all this work (and waiting) get us?

- Cheap emulated environments - emulate only the parts you need to
Multiarch: so what?

What does all this work (and waiting) get us?

- Cheap emulated environments - emulate only the parts you need to
- Cross-compilation is no longer special - you get it for free!
Multiarch: so what?

What does all this work (and waiting) get us?

- Cheap emulated environments - emulate only the parts you need to
- Cross-compilation is no longer special - you get it for free!
- Support for cross-grading a system from one architecture to another (arm→armel, i386→amd64, armel→armhf)
Multiarch: so what?

What does all this work (and waiting) get us?

- Cheap emulated environments - emulate only the parts you need to
- Cross-compilation is no longer special - you get it for free!
- Support for cross-grading a system from one architecture to another (arm → armel, i386 → amd64, armel → armhf)
- Better support for binary-only software
Multiarch: so what?

What does all this work (and waiting) get us?

- Cheap emulated environments - emulate only the parts you need to
- Cross-compilation is no longer special - you get it for free!
- Support for cross-grading a system from one architecture to another (arm → armel, i386 → amd64, armel → armhf)
- Better support for binary-only software ... the same support we’re already providing, now without impact on the supportability of the main archive
http://wiki.debian.org/Multiarch/Implementation gives all the details for converting shared libraries, the common interesting case

- use `dpkg-architecture -qDEB_HOST_MULTIARCH` to query the name of the subdirectory for libraries

- move shared libraries, static libraries, `.so` symlinks, pkg-config files, `.la` files (... if you still need them), and plugin loader paths to the new directory
  - plugin loader path changes obviously require coordination between related packages to manage the transition!
  - either Breaks, or searching both old and new paths

- leave helper executables in `/usr/lib` (the "libexec" case)
  - if the helper executables are binary, they need split into a separate package
  - policy says you’re already supposed to do this
packages that are dependencies of shared libraries because they ship data or executables (such as the above -bin, -run packages) must be marked Multi-Arch: foreign
▶ even if they’re Architecture: all!
▶ (see https://wiki.ubuntu.com/MultiarchSpec to understand why)
▶ (see previous slide about using written specs)
Multiarch: what lies ahead

- dev packages needed for cross-building
  - Easiest to move .so, .la, .a, .pc at the same time we move shared libraries
  - Policy allows this
  - But this is not enough to treat -dev packages as Multi-Arch: same
  - headers are sometimes architecture-dependent
  - we need to work through best practices for handling these, balancing package maintainability against avoidable filesystem bloat
- long list of libraries to convert before we can drop ia32-libs
- several short lists that let us start picking off the reverse-dependencies of ia32-libs (ping me if interested)
- get multiarch support merged into dpkg (in parallel to everything above)
- get downloading of i386 Packages files enabled by default on amd64!
- handful of bugs to sort out with eglibc
- make multiarch usable for upstreams
Multiarch: the next level

What we have in the archive already enables us to do lots of great things, but this is only the first of the "bite-sized" deliverables!

» Is it time to have "partial" architectures? (sparc64, ppc64, i686?)
» Do we need an amd64 grub-pc package, or should we make amd64 systems use the i386 one?
» Can we have sanely bootstrappable cross-compilers in the archive proper?
» Do we want to use cross-compiling by default on autobuilders? For bootstrapping new ports **directly** via the archive?
» Can we solve the use of Architecture: all packages in the archive that can only be built on one architecture? (E.g., firmware packages for emulators)
» The implementation is easy, the policy is hard
Multiarch: excelsior

The future of multiarch is wide open - and it’s up to you