Crosstoolchains in Debian

Wookey

The Cross-building victim
MultiarchCross

- Historical Context
- Toolchain build flavours
- Packaging for Jessie
- Current status
- Future craziness
Outline

1. Toolchain Basics
2. Toolchain Packaging
3. Current Packages
4. Future goodness
**Nomenclature**

**Build**: Machine/architecture you are building on

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

**Target**: Machine/architecture a compiler generates code for
Toolchain

There are 2 aspects to multiarching toolchains

- **System search paths**
  - path for libs and system headers (<> includes)
  - Previously /usr/include/ (native), /usr/<triplet>/include (cross)
  - Now always /usr/include/<triplet>:/usr/include/
  - Previously /usr/lib/ (native), /usr/<triplet>/lib (cross)
  - Now always /usr/lib/<triplet>:/usr/lib:/lib/<triplet>:/lib

- **Build mechanism**
  - Standalone libc6-armel-cross arch all
  - Multiarch Depend on libc6:armel, libgcc1:armel
Toolchain History

- 2003 - emdebian cross-toolchains (non MA)
- 2010 - linaro/ubuntu cross-toolchains (MA paths)
- 2012 - multiarch-built cross-toolchains (MA paths, MA built)
- 2013 - bare-metal cross-toolchains (MA irrelevant)
- 2014 - secretsauce.net cross-toolchains (MA paths, MA built)
- 2014 - cross-toolchains in archive (MA paths, MA built)
Bits and Bobs

Other things are needed for a smooth experience

- build-essential-<arch> packages
- <triplet>-pkg-config → pkg-config-crosswrapper
- toolchain defaults links (arm-linux-gnueabi-gcc → arm-linux-gnueabi-gcc-4.9)
- autoconf cache/cmake TOOLCHAIN file (in dpkg-cross)
- <triplet>-tools
- sbuild support for crossbuilding
- multiarched libraries and -dev packages
- multiarch foreign tools
Actually building

Build locally:

- sbuild --host <arch>

Best in a chroot:

- sbuild --host <arch> -d <distro> <package>_<<version>>

1. chroot into <distro> (clean) chroot
2. update/upgrade
3. dpkg --add-architecture <arch>
4. apt-get install crossbuild-essential-<arch> (configurable)
5. apt-get -a <arch> build-dep <package>
6. CONFIG_SITE=/etc/dpkg-cross/cross-config.<arch> dpkg-buildpackage -a <arch>
7. clean up after build (throw away chroot)
Outline

1. Toolchain Basics
2. Toolchain Packaging
3. Current Packages
4. Future goodness
### Cross-toolchain-base packaging

#### ‘3-stage’ bootstrap

<table>
<thead>
<tr>
<th>Stage</th>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linux stage1</td>
<td>linux-libc-dev headers</td>
</tr>
<tr>
<td>2</td>
<td>Binutils</td>
<td>Binutils</td>
</tr>
<tr>
<td>3</td>
<td>GCC stage1</td>
<td>Bare C-compiler</td>
</tr>
<tr>
<td>4</td>
<td>glibc stage1</td>
<td>Minimal libc</td>
</tr>
<tr>
<td>5</td>
<td>GCC stage2</td>
<td>C-compiler against eglibc</td>
</tr>
<tr>
<td>6</td>
<td>glibc stage2</td>
<td>Full libc build (without libselinux)</td>
</tr>
<tr>
<td>7</td>
<td>GCC stage3</td>
<td>All compilers</td>
</tr>
</tbody>
</table>
Multiarch Toolchain build

- Source: `linux` ⇒ `linux-libc-dev:armhf`
- Source: `binutils-cross` ⇒ `binutils-arm-linux-gnueabihf`
- Source: `gcc-4.9` ⇒ `libgcc1:armhf, libstdc++:armhf`
- Source: `cross-gcc-4.9-armhf` ⇒ `gcc-4.9-arm-linux-gnueabihf`

with deps on target arch pkgs=yes dpkg-buildpackage
```
--target-arch armhf -d -T control
```
with deps on target arch pkgs=yes dpkg-buildpackage
```
--target-arch armhf -b
```
Build-depends: gcc-4.9-source:all, libc6-dev:armhf, libgcc1:armhf,
libstdc++-dev:armhf
**Standalone Toolchain build**

- **Source:** `linux` ⇒ `linux-libc-dev-armhf-cross:all`
- **Source:** `binutils-cross` ⇒ `binutils-arm-linux-gnueabihf`
- **Source:** `gcc-4.9` ⇒ `libgcc1-armhf-cross:all, libstdc++-armhf-cross:all`
- **Source:** `cross-gcc-4.9-armhf` ⇒ `gcc-4.9-arm-linux-gnueabihf`
  
  dpkg-buildpackage --target-arch armhf -d -T control
  dpkg-buildpackage --target-arch armhf -b

  **Build-depend:** `gcc-4.9-source:all, libc6-dev-armhf-cross:all, libgcc1-armhf-cross:all, libstdc++-dev-armhf-cross:all`
Considerations - multiarch

- Only one libc, libstdc++, libgcc
- Simple gcc build using existing stuff, against deps
- Build against deps is normal case, bootstrap is exceptional
- Can be used for bootstrap (rebootstrap)
- Multiarch sync makes unstable uninstallable (gcc upload once/week)
- Needs multiarch-build-ready infrastructure
- Does not do multilib
- Works today
Considerations - multiarch

- Only one libc, libstdc++, libgcc
- Simple gcc build using existing stuff, against deps
- Build against deps is normal case, bootstrap is exceptional
- Can be used for bootstrap (rebootstrap)
- Multiarch sync makes unstable uninstallable (gcc upload once/week)
- Needs multiarch-build-ready infrastructure
- Does not do multilib
- Works today
- Makes Doko angry
Considerations - standalone

- 2 copies of libc, libstdc++, libgcc
  lets configure/linker find/use the wrong one
- 3-stage bootstrap builds stuff we already have
  (kernel headers, libc, libgcc, libstdc++)
- Slow, lots to break
- Can install and build kernels without multiarch sync
- Multiarch sync for package builds (libgcc1)
- Useful for new arch, not yet in archive
- No buildd/wanna-build changes needed
- Broken on Debian for last 2 years
- Not keen to maintain this
Possible 3rd way

- Split binutils out (done)
- Separate gcc build from libraries build (mostly done)
- Build cross-gcc standalone
- Toolchain-base generates -cross libraries using dpkg-cross
- Toolchain-base has foreign-arch build-deps
- Sbuild+wanna-build update, but not britney
- Avoids glibc/gcc dance
- Unstable toolchains remain installable
- Quick, simple builds
- Only slightly ugly
**Multilib vs Multiarch**

---

**build i386 binaries**

- i386-linux-gnu-gcc
- x86_64-linux-gnu-gcc -m32

**build armel binaries**

- armel-linux-gnueabi-gcc
- armhf-linux-gnueabihf -mfloat-abi=softfp

Current cross-gcc built with `DEB_CROSS_NO_BIARCH=yes`

What needs multilib?
Outline

1. Toolchain Basics
2. Toolchain Packaging
3. Current Packages
4. Future goodness

Wookey (Linaro)
Crosstoolchains in Debian
MiniDebconf, Cambridge, 2014
cross-binutils

Built for all released linux arches

- arm64 binutils-aarch64-linux-gnu
- armel binutils-arm-linux-gnueabi
- armhf binutils-arm-linux-gnueabihf
- i386 binutils-i586-linux-gnu
- mips binutils-mips-linux-gnu
- mipsel binutils-mipsel-linux-gnu
- mips64el binutils-mips64el-linux-gnuabi64
- powerpc binutils-powerpc-linux-gnu
- ppc64le binutils-powerpc64le-linux-gnu
- amd64 binutils-x86-64-linux-gnu

Right set?
cross-gcc

One source package per arch.
jessie linux non-x86 arches
Built on amd64

cpp, gcc, g++, gfortran (gccgo, gobj in p.d.o repo).

- arm64 cross-gcc-4.9-arm64
- armel cross-gcc-4.9-armel
- armhf cross-gcc-4.9-armhf
- mips cross-gcc-4.9-mips
- mipsel cross-gcc-4.9-mipsel
- powerpc cross-gcc-4.9-powerpc
- ppc64el cross-gcc-4.9-ppc64el

Right set?
cross-gcc-defaults

Exactly like gcc-defaults

<table>
<thead>
<tr>
<th>cross-gcc-4.9-armhf (arch any)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpp-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>gcc-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>g++-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>gfortran-4.9-arm-linux-gnueabihf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cross-gcc-defaults (arch all)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpp-arm-linux-gnueabihf ⇒ cpp-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>gcc-arm-linux-gnueabihf ⇒ gcc-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>g++-arm-linux-gnueabihf ⇒ g++-4.9-arm-linux-gnueabihf</td>
</tr>
<tr>
<td>gfortran-arm-linux-gnueabihf ⇒ gfortran-4.9-arm-linux-gnueabihf</td>
</tr>
</tbody>
</table>
crossbuild-essential

Built from **build-essential** source package

- crossbuild-essential-armhf
- crossbuild-essential-armel
- crossbuild-essential-mips . . .

Empty package depending on:

- gcc-< triplet >
- g++-< triplet >
- libc-dev:< arch >
- build-essential:native

Installed by default by **sbuild** when crossing
Obsoleted by multiarch/gcc-for-host in Jessie+1
Crosstoolchain Release Goal

So near and yet so far . . .

- **cross-binutils** uploaded April, in testing 22nd Sept.
- **sbuild** (*multiarch-build*) in testing 8th Oct
- **wanna-build** patches tested/done 15th Oct
- **cross-gcc** uploaded 22nd Oct
- rebuilt against testing gcc 4.9.1-19 29th Oct
- **cross-gcc-defaults** 24th Oct
- cross-gcc must wait for last gcc upload - will be late
- cross-gcc only builds in jessie, not stable
- wanna-build/britney updates needed
- **build-essential** went to **experimental** 25th Oct (trouble!)
Consistent Target Arch Specifier

### Consistent target arch env var

- binutils: \$TARGET
- gcc: \$DEB_GCC_TARGET or \$GCC_TARGET
- gdb: \$GDB_TARGET

Should be \$DEB_TARGET_ARCH everywhere

### Consistent dpkg-buildpackage usage

```bash
dpkg-buildpackage --target-arch <arch>
```

(sets \$DEB_TARGET_ARCH, in dpkg 1.17.17)

overrides DEB_TARGET_ARCH in env
dh-autoreconf is good

- Consensus for dh-autoreconf (and/or autotools-dev)
- http://wiki.debian.org/Autoreconf
- Remove loads of makework, permanently
- Actually build from source
- No more 60K packages with 1Mb autotools diffs!
Outline

1. Toolchain Basics
2. Toolchain Packaging
3. Current Packages
4. Future goodness
Co-installable toolchains

https://wiki.debian.org/CoinstallableToolchains
Currently not possible to install gcc:i386 and gcc:amd64 together

Currently

gcc-<ver> contains the native compiler

gcc-<ver>-<triplet> contains a cross-compiler

Proposed

gcc-<ver>-x86_64-linux-gnu

gcc-<ver>-i386-linux-gnu

gcc-<ver>-arm-linux-gnueabihf

Need some symlinks swapping in the packaging to work
Build dependency translation

Some build-deps change name when crossing

binutils → binutils-<triplet>
gcc-4.8 → gcc-4.8-<triplet>
pkg-config → pkg-config-<triplet>
g-ir-scanner → g-ir-scanner-<triplet>

6 possible solutions:
https://wiki.debian.org/CrossTranslatableBuildDeps
## Orthogonal toolchains

### Package Layout

<table>
<thead>
<tr>
<th>for amd64</th>
<th>for mips</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package:</strong> gcc-for-build</td>
<td><strong>Package:</strong> gcc-for-build</td>
</tr>
<tr>
<td><strong>Architecture:</strong> all</td>
<td><strong>Architecture:</strong> all</td>
</tr>
<tr>
<td><strong>Multi-Arch:</strong> foreign</td>
<td><strong>Multi-Arch:</strong> foreign</td>
</tr>
<tr>
<td><strong>Depends:</strong> gcc</td>
<td><strong>Depends:</strong> gcc</td>
</tr>
<tr>
<td><strong>Contents:</strong> empty</td>
<td><strong>Contents:</strong> empty</td>
</tr>
<tr>
<td><strong>Package:</strong> gcc-for-host</td>
<td><strong>Package:</strong> gcc-for-host</td>
</tr>
<tr>
<td><strong>Architecture:</strong> mips</td>
<td><strong>Architecture:</strong> amd64</td>
</tr>
<tr>
<td><strong>Multi-Arch:</strong> same</td>
<td><strong>Multi-Arch:</strong> same</td>
</tr>
<tr>
<td><strong>Depends:</strong> gcc-mips-linux-gnu</td>
<td><strong>Depends:</strong> gcc-x86-64-linux-gnu</td>
</tr>
<tr>
<td><strong>Contents:</strong> empty</td>
<td><strong>Contents:</strong> empty</td>
</tr>
</tbody>
</table>

Wookey (Linaro)

Crosstoolchains in Debian
 Orthogonal toolchains 2

<table>
<thead>
<tr>
<th>Package Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for amd64</strong></td>
</tr>
<tr>
<td>Package: gcc-mips-linux-gnu</td>
</tr>
<tr>
<td>Architecture: amd64</td>
</tr>
<tr>
<td>Multi-Arch: foreign</td>
</tr>
<tr>
<td>Contents:</td>
</tr>
<tr>
<td>/usr/bin/mips-linux-gnu-gcc</td>
</tr>
<tr>
<td>Package: gcc-x86-64-linux-gnu</td>
</tr>
<tr>
<td>Architecture: amd64</td>
</tr>
<tr>
<td>Multi-Arch: foreign</td>
</tr>
<tr>
<td>Depends: gcc</td>
</tr>
<tr>
<td>Contents: empty</td>
</tr>
</tbody>
</table>

| **for mips**  |
| Package: gcc-mips-linux-gnu |
| Architecture: mips |
| Multi-Arch: foreign |
| Depends: gcc |
| Contents: empty |

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Package: gcc-86-64-linux-gnu</td>
</tr>
<tr>
<td>Architecture: mips</td>
</tr>
<tr>
<td>Multi-Arch: foreign</td>
</tr>
<tr>
<td>Depends: gcc</td>
</tr>
<tr>
<td>Contents:</td>
</tr>
<tr>
<td>/usr/bin/x86_64-linux-gnu-gcc</td>
</tr>
</tbody>
</table>

wiki.debian.org/Sprints/2014/BootstrapSprint/Results
Source Build Depends?

Binary-source packages are a workaround
Build-depends: `binutils:src` nicer
(gcc-4.9-source is patched, gcc-4.9:source is not)

What would it take to fix?
What directory to install to?
Allow `apt-get source foo` during build?
Policy needs updating

- Multiarch is not described in policy.
  We need to fix that!
Thanks to various people

- **Linaro for funding this work**
- **Cross-toolchains team**
  - Dima Kogan (toolchain, sbuild, wanna-build fixes)
  - Thibaut Girka (multiarch cross-toochains)
  - Helmut Grohne (toolchain fixes, rebootstrap)
  - Agustin Henze, Thomas Preud’homme, Keith Packard (bare-metal)
- **Various useful people:** Johannes Schauer, Steve Langasek, Colin Watson, Marcin Juśkiewicz, Mattias Klose, Hector Oron, Neil Williams

Further reading: https://wiki.debian.org/CrossToolchains
Further reading: https://wiki.debian.org/Sprints/2014/BootstrapSprint/Results