

## NAME

dpkg-architecture – set and determine the architecture for package building

## SYNOPSIS

**dpkg-architecture** [*option...*] [*command*]

## DESCRIPTION

**dpkg-architecture** provides a facility to determine and set the build and host architecture for package building.

The build architecture is always determined by an external call to **dpkg**(1), and cannot be set at the command line.

You can specify the host architecture by providing one or both of the options **--host-arch** and **--host-type**. The default is determined by an external call to **gcc**(1), or the same as the build architecture if **CC** or **gcc** are both not available. One out of **--host-arch** and **--host-type** is sufficient, the value of the other will be set to a usable default. Indeed, it is often better to only specify one, because **dpkg-architecture** will warn you if your choice does not match the default.

## COMMANDS

**-l, --list**

Print the environment variables, one each line, in the format *VARIABLE=value*. This is the default action.

**-e, --equal *architecture***

Check for equality of architecture (since dpkg 1.13.13). It compares the current or specified Debian host architecture against *architecture*, to check if they are equal. This action will not expand the architecture wildcards. Command finishes with an exit status of 0 if matched, 1 if not matched.

**-i, --is *architecture-wildcard***

Check for identity of architecture (since dpkg 1.13.13). It compares the current or specified Debian host architecture against *architecture-wildcard* after having expanded it as an architecture wildcard, to check if they match. Command finishes with an exit status of 0 if matched, 1 if not matched.

**-q, --query *variable-name***

Print the value of a single variable.

**-s, --print-set**

Print an export command. This can be used to set the environment variables using eval.

**-u, --print-unset**

Print a similar command to **--print-unset** but to unset all variables.

**-c, --command *command-string***

Execute a *command-string* in an environment which has all variables set to the determined value.

**-L, --list-known**

Print a list of valid architecture names. Possibly restricted by one or more of the matching options **--match-wildcard**, **--match-bits** or **--match-endian** (since dpkg 1.17.14).

**–?, --help**

Show the usage message and exit.

**--version**

Show the version and exit.

## OPTIONS

**-a, --host-arch *architecture***

Set the host Debian architecture.

- t, --host-type** *gnu-system-type*  
Set the host GNU system type.
- A, --target-arch** *architecture*  
Set the target Debian architecture (since dpkg 1.17.14).
- T, --target-type** *gnu-system-type*  
Set the target GNU system type (since dpkg 1.17.14).
- W, --match-wildcard** *architecture-wildcard*  
Restrict the architectures listed by **--list-known** to ones matching the specified architecture wildcard (since dpkg 1.17.14).
- B, --match-bits** *architecture-bits*  
Restrict the architectures listed by **--list-known** to ones with the specified CPU bits (since dpkg 1.17.14). Either **32** or **64**.
- E, --match-endian** *architecture-endianness*  
Restrict the architectures listed by **--list-known** to ones with the specified endianness (since dpkg 1.17.14). Either **little** or **big**.
- f, --force**  
Values set by existing environment variables with the same name as used by the scripts are honored (i.e. used by **dpkg-architecture**), except if this force flag is present. This allows the user to override a value even when the call to **dpkg-architecture** is buried in some other script (for example **dpkg-buildpackage(1)**).

## TERMS

### build machine

The machine the package is built on.

### host machine

The machine the package is built for.

### target machine

The machine the compiler is building for. This is only needed when building a cross-toolchain, one that will be built on the build architecture, to be run on the host architecture, and to build code for the target architecture.

### Debian architecture

The Debian architecture string, which specifies the binary tree in the FTP archive. Examples: i386, sparc, hurd-i386.

### Debian architecture tuple

A Debian architecture tuple is the fully qualified architecture with all its components spelled out. This differs with Debian architectures in that at least the *cpu* component does not embed the *abi*. The current tuple has the form *abi-libc-os-cpu*. Examples: base-gnu-linux-amd64, eabihf-musl-linux-arm.

### Debian architecture wildcard

A Debian architecture wildcard is a special architecture string that will match any real architecture being part of it. The general form is a Debian architecture tuple with four or less elements, and with at least one of them being **any**. Missing elements of the tuple are prefixed implicitly as **any**, and thus the following pairs are equivalent:

**any-any-any-any = any**

**any-any-os-any = os-any**

**any-libc-any-any = libc-any-any**

Examples: linux-any, any-i386, hurd-any, eabi-any-any-arm, musl-any-any.

### GNU system type

An architecture specification string consisting of two parts separated by a hyphen: *cpu* and *system*. Examples: i586-linux-gnu, sparc-linux-gnu, i686-gnu, x86\_64-netbsd.

**multiarch triplet**

The clarified GNU system type, used for filesystem paths. This triplet does not change even when the baseline ISA gets bumped, so that the resulting paths are stable over time. The only current difference with the GNU system type is that the CPU part for i386 based systems is always i386. Examples: i386-linux-gnu, x86\_64-linux-gnu. Example paths: /lib/powerpc64le-linux-gnu/, /usr/lib/i386-kfreebsd-gnu/.

**VARIABLES**

The following variables are set by **dpkg-architecture** (see the **TERMS** section for a description of the naming scheme):

**DEB\_BUILD\_ARCH**

The Debian architecture of the build machine.

**DEB\_BUILD\_ARCH\_ABI**

The Debian abi name of the build machine (since dpkg 1.18.11).

**DEB\_BUILD\_ARCH\_LIBC**

The Debian libc name of the build machine (since dpkg 1.18.11).

**DEB\_BUILD\_ARCH\_OS**

The Debian system name of the build machine (since dpkg 1.13.2).

**DEB\_BUILD\_ARCH\_CPU**

The Debian cpu name of the build machine (since dpkg 1.13.2).

**DEB\_BUILD\_ARCH\_BITS**

The pointer size of the build machine (in bits; since dpkg 1.15.4).

**DEB\_BUILD\_ARCH\_ENDIAN**

The endianness of the build machine (little / big; since dpkg 1.15.4).

**DEB\_BUILD\_GNU\_CPU**

The CPU part of **DEB\_BUILD\_GNU\_TYPE**.

**DEB\_BUILD\_GNU\_SYSTEM**

The System part of **DEB\_BUILD\_GNU\_TYPE**.

**DEB\_BUILD\_GNU\_TYPE**

The GNU system type of the build machine.

**DEB\_BUILD\_MULTIARCH**

The clarified GNU system type of the build machine, used for filesystem paths (since dpkg 1.16.0).

**DEB\_HOST\_ARCH**

The Debian architecture of the host machine.

**DEB\_HOST\_ARCH\_ABI**

The Debian abi name of the host machine (since dpkg 1.18.11).

**DEB\_HOST\_ARCH\_LIBC**

The Debian libc name of the host machine (since dpkg 1.18.11).

**DEB\_HOST\_ARCH\_OS**

The Debian system name of the host machine (since dpkg 1.13.2).

**DEB\_HOST\_ARCH\_CPU**

The Debian cpu name of the host machine (since dpkg 1.13.2).

**DEB\_HOST\_ARCH\_BITS**

The pointer size of the host machine (in bits; since dpkg 1.15.4).

**DEB\_HOST\_ARCH\_ENDIAN**

The endianness of the host machine (little / big; since dpkg 1.15.4).

**DEB\_HOST\_GNU\_CPU**

The CPU part of **DEB\_HOST\_GNU\_TYPE**.

**DEB\_HOST\_GNU\_SYSTEM**

The System part of **DEB\_HOST\_GNU\_TYPE**.

**DEB\_HOST\_GNU\_TYPE**

The GNU system type of the host machine.

**DEB\_HOST\_MULTIARCH**

The clarified GNU system type of the host machine, used for filesystem paths (since dpkg 1.16.0).

**DEB\_TARGET\_ARCH**

The Debian architecture of the target machine (since dpkg 1.17.14).

**DEB\_TARGET\_ARCH\_ABI**

The Debian abi name of the target machine (since dpkg 1.18.11).

**DEB\_TARGET\_ARCH\_LIBC**

The Debian libc name of the target machine (since dpkg 1.18.11).

**DEB\_TARGET\_ARCH\_OS**

The Debian system name of the target machine (since dpkg 1.17.14).

**DEB\_TARGET\_ARCH\_CPU**

The Debian cpu name of the target machine (since dpkg 1.17.14).

**DEB\_TARGET\_ARCH\_BITS**

The pointer size of the target machine (in bits; since dpkg 1.17.14).

**DEB\_TARGET\_ARCH\_ENDIAN**

The endianness of the target machine (little / big; since dpkg 1.17.14).

**DEB\_TARGET\_GNU\_CPU**

The CPU part of **DEB\_TARGET\_GNU\_TYPE** (since dpkg 1.17.14).

**DEB\_TARGET\_GNU\_SYSTEM**

The System part of **DEB\_TARGET\_GNU\_TYPE** (since dpkg 1.17.14).

**DEB\_TARGET\_GNU\_TYPE**

The GNU system type of the target machine (since dpkg 1.17.14).

**DEB\_TARGET\_MULTIARCH**

The clarified GNU system type of the target machine, used for filesystem paths (since dpkg 1.17.14).

**FILES****Architecture tables**

All these files have to be present for **dpkg-architecture** to work. Their location can be overridden at runtime with the environment variable **DPKG\_DATADIR**. These tables contain a format **Version** pseudo-field on their first line to mark their format, so that parsers can check if they understand it, such as "# Version=1.0".

*/usr/share/dpkg/cputable*

Table of known CPU names and mapping to their GNU name. Format version 1.0 (since dpkg 1.13.2).

*/usr/share/dpkg/ostable*

Table of known operating system names and mapping to their GNU name. Format version 2.0 (since dpkg 1.18.11).

*/usr/share/dpkg/tupletable*

Mapping between Debian architecture tuples and Debian architecture names. Format version 1.0 (since dpkg 1.18.11).

*/usr/share/dpkg/abitable*

Table of Debian architecture ABI attribute overrides. Format version 2.0 (since dpkg 1.18.11).

### Packaging support

*/usr/share/dpkg/architecture.mk*

Makefile snippet that properly sets and exports all the variables that **dpkg-architecture** outputs (since dpkg 1.16.1).

### EXAMPLES

**dpkg-buildpackage** accepts the **-a** option and passes it to **dpkg-architecture**. Other examples:

```
CC=i386-gnu-gcc dpkg-architecture -c debian/rules build
```

```
eval `dpkg-architecture -u`
```

Check if the current or specified host architecture is equal to an architecture:

```
dpkg-architecture -elinux-alpha
```

```
dpkg-architecture -amips -elinux-mips
```

Check if the current or specified host architecture is a Linux system:

```
dpkg-architecture -ilinux-any
```

```
dpkg-architecture -ai386 -ilinux-any
```

### Usage in debian/rules

The environment variables set by **dpkg-architecture** are passed to *debian/rules* as make variables (see make documentation). However, you should not rely on them, as this breaks manual invocation of the script. Instead, you should always initialize them using **dpkg-architecture** with the **-q** option. Here are some examples, which also show how you can improve the cross compilation support in your package:

Retrieving the GNU system type and forwarding it to *./configure*:

```
DEB_BUILD_GNU_TYPE ?= $(shell dpkg-architecture -qDEB_BUILD_GNU_TYPE)
DEB_HOST_GNU_TYPE ?= $(shell dpkg-architecture -qDEB_HOST_GNU_TYPE)
[...]
ifeq ($(DEB_BUILD_GNU_TYPE), $(DEB_HOST_GNU_TYPE))
    confflags += --build=$(DEB_HOST_GNU_TYPE)
else
    confflags += --build=$(DEB_BUILD_GNU_TYPE) \
                --host=$(DEB_HOST_GNU_TYPE)
endif
[...]
./configure $(confflags)
```

Doing something only for a specific architecture:

```
DEB_HOST_ARCH ?= $(shell dpkg-architecture -qDEB_HOST_ARCH)

ifeq ($(DEB_HOST_ARCH),alpha)
    [...]
endif
```

or if you only need to check the CPU or OS type, use the **DEB\_HOST\_ARCH\_CPU** or **DEB\_HOST\_ARCH\_OS** variables.

Note that you can also rely on an external Makefile snippet to properly set all the variables that **dpkg-architecture** can provide:

```
include /usr/share/dpkg/architecture.mk

ifeq ($(DEB_HOST_ARCH),alpha)
    [...]
```

endif

In any case, you should never use **dpkg --print-architecture** to get architecture information during a package build.

## ENVIRONMENT

### DPKG\_DATADIR

If set, it will be used as the **dpkg** data directory, where the architecture tables are located (since dpkg 1.14.17). Defaults to «/usr/share/dpkg».

### DPKG\_COLORS

Sets the color mode (since dpkg 1.18.5). The currently accepted values are: **auto** (default), **always** and **never**.

### DPKG\_NLS

If set, it will be used to decide whether to activate Native Language Support, also known as internationalization (or i18n) support (since dpkg 1.19.0). The accepted values are: **0** and **1** (default).

## NOTES

All long command and option names available only since dpkg 1.17.17.

## SEE ALSO

**dpkg-buildpackage(1)**.