Packages not using the default build flags: a taxonomy

Emanuele Rocca
DebConf25, July 2025

Some packages
are not using the default flags
even though they should

Outline

Build flags in Debian

Packages that should use them but don’t

Why not?

Build flags in Debian

Build Flags

Hello World!

Behavior of compiler, linker, assembler,... can be controlled via command-line arguments, or `build flags`.

gcc -Wall -O2 -g hello.c -o hello

Build Flags

Why changing the behavior of compiler and linker?

- Security features
- Performance tuning
- Reproducible builds
- Adherence to standards
- How to deal with warnings and errors

arm

Build Flags

Environment variables, see GNU Make manual

- CFLAGS
- CPPFLAGS
- CXXFLAGS
- FFLAGS
- ASFLAGS
- LDFLAGS

arm

Default Build Flags in Debian

Which build flags should programs in Debian use?

- 2008: Introducing security hardening features for lenny
- 2011: dpkg-buildflags returns hardening build flags by default

arm

Default Build Flags in Debian

Get and set build flags: dpkg-buildflags(1)

- Flags can be customized system-wide, eg. for mass rebuilds: `/etc/dpkg/buildflags.conf`
- For a given package by maintainers: `DEB_CFLAGS_MAINT_APPEND`, `DEB_CFLAGS_MAINT_SET`...
- For a given package by users: `DEB_CFLAGS_APPEND`, `DEB_CFLAGS_SET`... Never use those in `debian/rules`!

Packages that should use them but don’t

PAC/BTI

How I got nerdsniped

- ARM64 Security features designed to mitigate control flow attacks
- Build flag **-mbranch-protection=standard** enables them
- Added to the default `CFLAGS` via `dpkg-buildflags`
- Binaries built with PAC/BTI include a special ELF note
- Not all packages get the feature!

arm

ELF files analysis

- Arch-indep packages must be excluded
- Not all binary-dep packages ship ELF files
- Not all ELF files are produced by compilers supporting the flag! (Haskell, Go, OCaml, Rust, Pascal, Lisp, ...)

Archive analysis

- Analyze the archive using **snapshot.debian.org**: time machine for the Debian archive
- JSON API: list of snapshosts like 20250109T024634Z, 20250109T084424Z ...
  `http://snapshot.debian.org/mr/timestamp/?after=2025-01-09`
- Download all arch-dependent debs found on
  http://snapshot.d.o/archive/debian/$TS/dists/sid/main/binary-arm64/Packages.gz
- `dpkg --extract $deb`, for each ELF file
- `readelf --notes $f | grep -q 'AArch64 feature:  BTI, PAC'`
- Downloading the archive takes about 20 minutes
- Running the analysis with `parallel` on a 8 core system takes 30 minutes

Analysis results

- 71576 binary packages
- 36008 arch-dependent (50% of total)
- 24233 ship ELFs (30% of total)
- About 600 source packages should get PAC/BTI, but don’t

Why not?

Main causes

- Hand-written Makefiles (hello scientists!)
- Misconfigured build systems
- Ancient debhelper usage
- Flags (in)voluntarily hardcoded in `debian/rules`

Revenge for Kerberos!

Hand-written Makefile Example

```
1 CFLAGS=-Wall -g -DI_AM_A_SUCKER
2
3 ccc.o: ccc.cc texmaker.o stringlist.o bytevector.o
4   $(CC) -c ccc.cc $(CFLAGS)
```

Better: `Conditional Variable Assignment` operator (`CFLAGS ?= -Wall`)

`https://sources.debian.org/src/cdcover/0.9.1-14/texmaker.cc/#L73`

arm

Misconfigured build system

CMake

```
1 set(CMAKE_C_FLAGS "-Wall -O3 -pthread")
```

Better:

```
1 set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O3 -pthread")
```

Ancient debhelper usage

- `dh(1)` introduced 17 years ago (2008)
- Since compat level 9 (2012) `dh` automatically sets all environment variables produced by `dpkg-buildflags`
- A few packages use an ancient compatibility level
- Others don't adopt the `dh` short style format
- Need to manually include `/usr/share/dpkg/buildflags.mk`

Flags voluntarily hardcoded

Maintainer explicitly sets the build flags manually in debian/rules

Flags involuntarily hardcoded

On line 3, `CFLAGS` is not set yet. debhelper only sets it later on, if not set yet! Doh.

```
1 #!/usr/bin/make -f
2
3 export CFLAGS += -pipe -fPIC -Wall
4
5 %:
6     dh $@
```

Solution: use `DEB_CFLAGS_MAINT_APPEND` instead.

Conclusions

- `dpkg-buildflags` provides an excellent mechanism for setting build flags
- Packages doing Normal Things (TM) are fine
- ”I’ll just write a couple of Makefiles myself” - Statements dreamed up by the utterly deranged
- Long tail of a few hundred packages to fix
- Fixes are generally easy!

blhc or similar

Just grep the build logs, right?

- Checking if a flag is NOT used at all is simple (grep -v)
- Knowing if it was supposed to be used is hard

arm