Reproducible builds

It is generally desirable that building the same source code with the same set of tools is reproducible, i.e. the output is always exactly the same. This makes it possible to verify that the build infrastructure for a binary distribution or embedded system has not been subverted. This can also make it easier to verify that a source or tool change does not make any difference to the resulting binaries.

The Reproducible Builds project has more information about this general topic. This document covers the various reasons why building the kernel may be unreproducible, and how to avoid them.

Timestamps

The kernel embeds timestamps in three places:

- The version string exposed by `uname()` and included in `/proc/version`
- File timestamps in the embedded initramfs
- If enabled via `CONFIG_IKHEADERS`, file timestamps of kernel headers embedded in the kernel or respective module, exposed via `/sys/kernel/kheaders.tar.xz`

By default the timestamp is the current time and in the case of `kheaders` the various files' modification times. This must be overridden using the KBUILD_BUILD_TIMESTAMP variable. If you are building from a git commit, you could use its commit date.

The kernel does *not* use the `__DATE__` and `__TIME__` macros, and enables warnings if they are used. If you incorporate external code that does use these, you must override the timestamp they correspond to by setting the SOURCE_DATE_EPOCH environment variable.

User, host

The kernel embeds the building user and host names in `/proc/version`. These must be overridden using the KBUILD_BUILD_USER and KBUILD_BUILD_HOST variables. If you are building from a git commit, you could use its committer address.

Absolute filenames

When the kernel is built out-of-tree, debug information may include absolute filenames for the source files. This must be overridden by including the `-fdebug-prefix-map` option in the KCFLAGS variable.

Depending on the compiler used, the `__FILE__` macro may also expand to an absolute filename in an out-of-tree build. Kbuild automatically uses the `-fmacro-prefix-map` option to prevent this, if it is supported.

The Reproducible Builds web site has more information about these prefix-map options.

Generated files in source packages

The build processes for some programs under the `tools/` subdirectory do not completely support out-of-tree builds. This may cause a later source package build using e.g. `make rpm-pkg` to include generated files. You should ensure the source tree is pristine by running `make mrproper` or `git clean -d -f -x` before building a source package.

Module signing

If you enable `CONFIG_MODULE_SIG_ALL`, the default behaviour is to generate a different temporary key for each build, resulting in the modules being unreproducible. However, including a signing key with your source would presumably defeat the purpose of signing modules.

One approach to this is to divide up the build process so that the unreproducible parts can be treated as sources:

1. Generate a persistent signing key. Add the certificate for the key to the kernel source.
2. Set the `CONFIG_SYSTEM_TRUSTED_KEYS` symbol to include the signing key's certificate, set `CONFIG_MODULE_SIG_KEY` to an empty string, and disable `CONFIG_MODULE_SIG_ALL`. Build the kernel and modules.
3. Create detached signatures for the modules, and publish them as sources.
4. Perform a second build that attaches the module signatures. It can either rebuild the modules or use the output of step 2.

Structure randomisation

If you enable `CONFIG_GCC_PLUGIN_RANDSTRUCT`, you will need to pre-generate the random seed in `scripts/gcc-plugins/randomize_layout_seed.h` so the same value is used in rebuilds.

Debug info conflicts

This is not a problem of unreproducibility, but of generated files being *too* reproducible.

Once you set all the necessary variables for a reproducible build, a vDSO's debug information may be identical even for different kernel versions. This can result in file conflicts between debug information packages for the different kernel versions.

To avoid this, you can make the vDSO different for different kernel versions by including an arbitrary string of "salt" in it. This is specified by the Kconfig symbol `CONFIG_BUILD_SALT`.