Sub-packages, dependencies and information flow

The case of the Autotools, Jitter and Poke

Luca Saiu
positron@gnu.org
http://ageinghacker.net
GNU Project

GNU Hackers' Meeting 2019
Madrid, Spain
September 6th 2019

The master copy is at http://ageinghacker.net/talks/ .

The code described here is published at http://ageinghacker.net/talks/nesting-sources--saiu--ghm-2019--2019-09-06.tar.gz .

Introduction

Poke, by José Marchesi, relies on Jitter, by me.

- Poke includes a compiler, generating code (at run time) for a VM generated (ahead of time) by Jitter
- Jitter is a dependency of Poke's
  - José's idea: distribute Jitter as part of Poke, *à-la*-Gnulib...
    - ... making Jitter a sub-package instead of a dependency.

A simplified example

The Automake manual contains a nice example:

- `arm/` is program, requiring...
- ...`hand/`, a library

The `hand/` source directory is contained within the `arm/` source directory, as `arm/hand/` so that `hand/` is a *sub-package* of `arm/`'s.
Both use Autoconf and Automake.

I expanded the example, writing C files and adding a `configure`-time option in `hand/`, to show you my point.

Demo, first version

[Demo]

Nesting, dependencies and information flow

Three questions about `arm` and `arm/hand`:

- which package depends on the other, at build time? (Obvious)
- which package depends on the other, at configuration time?
- which `configure` script passes information to the other? How does information flow between `arm` and `arm/hand`?

What if arm/ depended on arm/hand/'s configuration?

Imagine that arm/ needs to base some of its configure-time decisions on the configuration of hand/.

That is the case of Poke with Jitter and, I anticipate, of many other projects which could benefit from basing their configure- or make-time behavior on a sub-package's configure-time choices, if it were made easier.

This *information flowing* from the sub-package's configuration *back to the super-package configuration* does not fit AC_CONFIG_SUBDIRS as presented in the Autoconf and Automake manuals: configuring sub-packages at the time of AC_OUTPUT is too late!

Demo, second version

[Demo]

How to make this reusable

Nothing difficult: an Autoconf macro. It will be distributed along with Jitter in `jitter.m4`. [Thanks to Darshit Shah for suggesting *the GNU Autoconf Archive*. I will contact the Autoconf maintainers as well.]

Then, it will be usable from the super-package's `configure.ac` like any other Autoconf macro:

In `arm/configure.ac`

```
AC_CONFIG_SUBDIRS_NOW([hand])
```

(Likely with another name for the macro so as not to conflict with Autoconf's namespace, unless it becomes official.)

Demo, last version

[Demo]

The macro definition

In `jitter.m4`, for example

```
AC_DEFUN([AC_CONFIG_SUBDIRS_NOW], [
ac_config_files_backup="$ac_config_files"
ac_config_files=''
subdirs_backup="$subdirs"
subdirs=''
AC_CONFIG_SUBDIRS([$1])
AC_OUTPUT
rm config.status
ac_config_files="$ac_config_files_backup"
subdirs="$subdirs_backup"
])
```

The macro definition — possible minor refinements

In `jitter.m4`, for example

```
AC_DEFUN([AC_CONFIG_SUBDIRS_NOW], [
ac_config_files_backup="$ac_config_files"
ac_config_files=''
subdirs_backup="$subdirs"
subdirs=''
AC_CONFIG_SUBDIRS([$1])
_AC_OUTPUT_SUBDIRS   # An internal macro.
                     # No need to delete config.status here.
ac_config_files="$ac_config_files_backup"
subdirs="$subdirs_backup"
])
```

What do you think?

Would you use this feature? Be it internally clean or not I find it useful, so I will make it available. The core macro described here is self-contained and independent from Jitter.

I am particularly interested in feedback from Autoconf experts about any weakness of this simple solution.

Thanks.