Configuring build base on FreeBSD

Making things easy for the user

FreeBSD has some knobs to set in order to build the base system or to avoid building some parts of the system. This way the build process could be optimized to avoid wasting time.

The current way of doing this is to edit the files src.conf(5) and src-env.conf(5), and the man pages provide information about how to do that. They also provide information about the dependencies of each of the knobs.

This aproach is not thought for a person that just jumped into FreeBSD, and trying to understand everything could be quite complicated. The main point of this paper is to allow a beginner in FreeBSD to configure the build process of the base system using a user-friendly interface based on dialog(1) and check the dependencies of the selected knobs in order to avoid the user wasting time in building something with some non-desired or without some desired components.

Design of the interface

At first, I made the same mistake that most of new developer do, try to do things fast and not taking care of the design, aiming it for next iterations. Due to personal duties, I could not properly finish it and when I retook it, it did not work (what a surprise!). So I design it from the beginning and avoid wasting too much time in the future.

We start by saying that we only allow one configuration for a src.conf(5) file at the same time, therefore, we have a mutex.

I have made it simple but working, so the presented algorithm displayed on Figure 1, shows how it was designed.

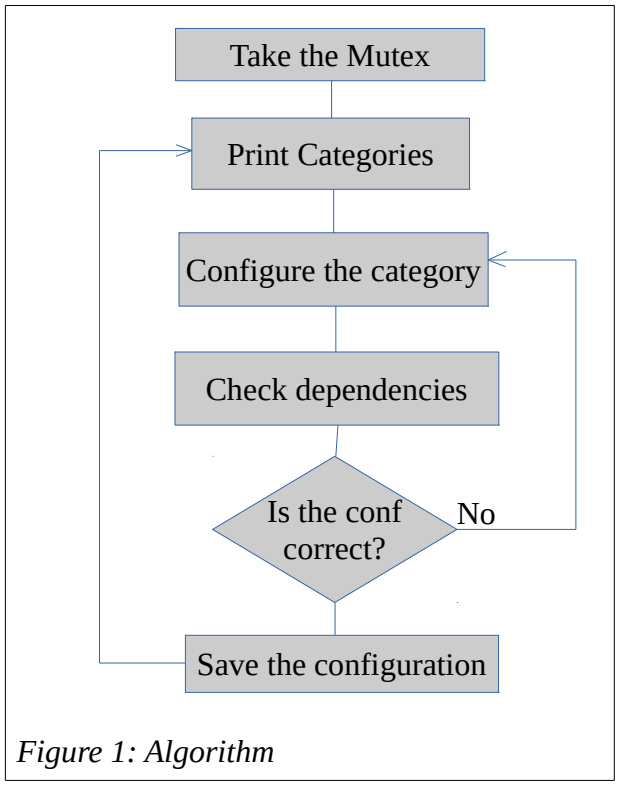

Figure 1: Algorithm

We take a mutex, we present the first message to the user, and she choose which category of knobs she would like to configure. Once she has finished selecting the options to configure, the dependencies of the selected knobs are checked.

If the user has selected a combination of knobs that has some dependencies which have not been selected, the user will be warned and she will stay at the current screen until she choose a valid configuration. Otherwise, she will get into the first dialog.

Once she has finished configuring all the knobs desired for her system, she can choose to save the current configuration or revert it.

Implementation of the algorithm

We have talked about the design, but without an implementation, it will be useless.

The first thing to do is define which are the options that we should present the user to choose. So I came with the following categories: admin, devel, docs, drivers, emulation, network, other, programs and security.

Under admin will be put all the programs that can be used to administrate a system like bsdinstall(8).

Under devel will be presented all the options that are usefull for the build or develop process of the machine, like which compiler should be used to build the new system.

The network category allows the user to configure the connectivity of the machine, for instance, if there should be support for Ipv6 in the final machine or if ipfw(8) should be installed on the machine.

Emulation is the simplest one, whether bhyve(8) or NDIS support should be integrated on the system.

Under programs, the options allows to choose which version of programs do we want in our own system, like GNU grep(1) instead of BSD grep(1).

The favorite of everyone, security, includes the cryptographic interface on OpenSSH, OpenSSL and the Capability system in FreeBSD (CAPSICUM).

The divers section collects the driver build for FreeBSD such as the floppy support (is it usefull nowadays to be active by default?) or the ZFS module.

Under others, we put everything that do not match in any of the other categories. It could be the boot loader type to use (UEFI) or if we want or system with forth.

And at last, but not least, the docs category compiles all the knobs that have something to do with the documentation of the project.

All this categories and descriptions and so on are been put under share/mk/src.opts.desc.mk, so they will be easily collected by the main Makefile (Makefile.config) and presented to the user.

Inside this file, Makefile.config, we define one config-${CATEGORY} target for each of the categories, allowing the user to configure it manually. Beside this, there are also some other targets, like check-config or config-save, but the most important ones are the following three: config-before, config-current and config-default.

The first one is a collection of the situation before the script, the second is the configuration on the current status and the last is the configuration without no knob defined.

This final one avoid the writing of default knobs on the src.conf(5) file, saving some space in case that it will be needed (doubt it).

The dependency check is controlled by the difference between config-before and config-current, and when they do not match the knobs that will be forced to some value are presented to the user as a dialog message, so she can change the right option and it will be checked again.

There is also the posibility of removing a current configuration with the rmconfig target, and this will revert the configuration to the default one.

Future work

The future work includes the revision of the description for the knobs, which right now it is inexistent and the posibility of configure the options and devices of the kernel though a similar interface. And checking if the kernel could be built before going into the whole compilation process.

A better documentation for each of the knobs can be written directly on the code, so the user gets more information about how it works.

A possible future work in this aspect could be to add in dialog(1) the possibility to display dependant lists of options and select directly live, the dependant options for each knob.