1 Introduction

There are two kinds of options for each module :
– inclusive (check boxes),
– exclusives (radio boxes), only one choice per section.

For each of these options, there can be a text box that allows to specify a value for that option, and even provide a default value. The only condition for this is to append “ :” to the name of that option.

There is for instance : `--marcel-stacksize:64` (64 being the default value).

The following files should be put in `module-name/config/options` to define an option :

1. `nnnoption-name:.sh` action of the option
2. `nnnoption-name:.help` provides help on the option
3. `nnnoption-name:.dep` specifies eventual dependencies
4. `nnnoption-name:.dft` specifies a default value

`nnn` is the number of the option and is used for sorting them. It should start with a 0 for inclusive options, and 1 for exclusive options.

Help files just contain the text that should be disabled *e.g.* in popups in `ezflavor`.

After adding/supressing/modifying options, `pm2-recreate-links` has to be called to take them into account.

2 Actions

A flavor is just a concatenation of all the `.sh` files of all the options that have been selected. These are shell script snippets that can do anything they need. Their eventual effect is to modify some environment variables. In the lists below, `${MOD}` is the module name in capital letters. The `appli` module corresponds to the application itself.

Read-only variables
– `PM2_ARCH` is set to the target architecture. See `bin/pm2-arch` for a list.
– `PM2_SYS` is set to the target architecture. See `bin/pm2-sys` for a list.
– `PM2_${MOD}_BUILD_DYNAMIC` is set to `yes` by the `build_dynamic` option to request compiling the module as a shared library.
– `PM2_${MOD}_BUILD_STATIC` is set to `yes` by the `build_dynamic` option to request compiling the module as a static library.
– `PM2_${MOD}_CFLAGS` contains the C flags that will be used while compiling all modules.
– `PM2_${MOD}_CFLAGS_KERNEL` contains the C flags that will be used while compiling module `${MOD}`.
– `PM2_${MOD}_LD_PATH` contains `-L` options that will be passed to the linker when linking a shared version of the module, or when statically linking the module into an application.
– `PM2_${MOD}_LIBS` contains `-l` options that will be passed to the linker when linking a shared version of the module, or when statically linking the module into an application.
– `PM2_${MOD}_EARLY_LDFLAGS` contains linker flags that will be passed before objects.
– `PM2_${MOD}_EARLY_LDFLAGS_KERNEL` contains linker flags that will be passed before objects, but only when linking the module itself.
– `PM2_${MOD}_DYN_LIBS_DIR` contains a list of directories which will be added to `LD_LIBRARY_PATH` by `pm2-load`.

- PM2_${MOD}_MAKEFILE contains text that will be pasted as is in the module's Makefile.
- PM2_${MOD}_LIBNAME overrides the name of the produced library (which is the name of the module by default).
- PM2_${MOD}_LINK_OTHERS should be set to yes if the module needs to be linked against all other modules, so that the module's shared library will pull all the others at runtime. This is notably needed when building ABI-compatibility libraries like libpthread.so.
- PM2_COMMON_PM2_SHLIBS contains the list of modules that are compiled as shared libraries.
- PM2_COMMON_FORTRAN_TARGET specifies which fortran compiler should be used, if any.
- PM2_CC specifics which C compiler should be used.
- PM2_PROTOCOLS contains the list of Madeleine protocols that will be available.
- PM2_NMAD_DRIVERS contains the list of Madeleine drivers.
- PM2_NMAD_INTERFACES contains the list of Madeleine interfaces.
- PM2_NMAD_LAUNCHER specifies which launder Madeleine should use.
- PM2_NMAD_STRATEGIES contains the list of Madeleine strategies.
- PM2_NMAD_STRAT specifies which Madeleine strategy should be used.
- PM2_LOADER specifies which loader should be run by pm2-load.

The following variables don't seem to have effect any more.

- PM2_${MOD}_MODULE_DEPEND_LIB was probably used for inter-module dependencies.

The following functions are provided (see bin/pm2-config-tools.sh).

- included_in what where looks for the word what in the string where.
- not_included_in does the contrary.
- defined_in what where looks for the word -Dwhat in the string where.
- not_defined_in does the contrary.

3 Dependencies

Some options require that others be set. For instance, module debug support depends on the debug part of TBX. This is expressed in the .dep files : tbx/config/options/00debug.dep has a Provide: TBX_debug line, and */config/options/00debug.dep contain a Depend: TBX_debug line. The name of the dependency, TBX_debug, but should be prefixed by the name of the module that provides it. Conflicts can also be expressed by putting a Conflict: line in the .dep file. The special case of depending on a module itself is handled by just using Depend: name.

4 Generic options

A few options are common to all modules : debugging flags, linking options, etc. generic/config/options contains templates for these. Modifications thus need to be done there, and make duplique_global run from there to propagate the modification to all modules.

5 Common options

A few options are not specific to any modules, they live in common/config/options.

6 Handling flavors

To configure flavors, either use `ezflavor`, `make config`, or use the following commands :

- `pm2-flavor get --flavor=flavor-name` : Shows the flavor.
- `pm2-flavor list` : Give the list of flavors.
- `pm2-flavor set --flavor=flavor-name` : Reset the whole flavor.
- ``pm2-flavor set --flavor=flavor-name `cat FILE` `` : Set flavor options from a file.
- `xargs pm2-flavor set --flavor=flavor-name < FILE` : Other way to achieve the same.
- `pm2-create-sample-flavor flavor-name` : Generate one of the sample flavors.