PTXdist Application Note

Working with Sources

PTXdist supports integrating your own sources into the build. It handles the correct toolchain to be used, handles dependencies to other packages your sources are require and also selecting correct pathes for include files and libraries. This application note describes the steps to do to make PTXdist work with your own sources.

Integration

PTXdist supports the building and maintaining of root filesystem creation. But everything is based on ready to use source file packets, mostly from the internet.
But also most projects require additional software development. This part describes a way to integrate our "work in progress" source files into PTXdist's build scheme, so we can test, debug and use them as any other packet on our target.

Steps to do:

1. creating a directory where our development should take place.
2. creating a development makefile in this directory that controls building the executable files or any other data from the sources we will later add.
3. creating an additional rule file in our own PTXdist project that conforms to the building rules of PTXdist but also controls our development makefiles and the files it generates.
4. activating the new rule file

The first step is simple. We can locating this directory anywhere we want or must (due to IT contraints or the using of software revision control systems).

The second step depends on our development environment. Many integrated development environments (IDE) currently in use supporting Makefile managing. This is especially valid for KDevelop and Eclipse but also for many others. So this development makefile will be autogenerated by the IDE. There are no constraints about this development makefile, because PTXdist's rule file we will add in the next step will act as an agent to handle all particularities of the development makefile.

But the simplest way to get a development makefile is to write it by our own.

```
#
# Simple development makefile
#
.PHONY: all install clean

all: my_executable

CFLAGS=-Werror -Wall

my_executable: my_source.c

clean:
rm -f my_executable

install:

#
# end of development makefile
#
```

Note: You can find this development makefile template in `Makefile` in PTXdist's

`<ptxdist-installdir>/rules/templates/kconfig/`

directory.

This works due to `make` has many built-in rules how to convert various source files into an executable files. We should keep our development makefile as simple as possible and we should also keep all internal `make` rules. This simplifies overwriting some rules and other macros from PTXdist's build environment to support a seamless cross development.

The more we add our own build rules the more PTXdist's rules file (maybe) must convert between both "worlds". It is sometimes tricky to build sources native to test on the host first and cross to run it on the target later. PTXdist supports such "tricks" and the simpler our development makefile is the more we can make use of these tricks.
The main goal for our source files should be to write code that runs unmodified on any architecture and host environment (and avoid any #ifdef...#endif hell as possible).

Creating an additional rule file for PTXdist's build environment into our active project is divided into two parts. First we create this rule file itself from a template into project's `rules/` directory:

```
$ cd rules
$ ptxdist newpacket source

ptxdist: creating a new packet from template-src:

ptxdist: enter packet name.......: my-packet
ptxdist: enter version number....:
ptxdist: enter URL of basedir....: file://home/jbe
ptxdist: enter packet author.....:
ptxdist: enter suffix............:
```

Note: In this case the prefix `file://` is required to tell PTXdist's download mechanism, this packet don't need to be downloaded, because its already at its place.

Note: Don't use other characters than [a-z0-9.+-] in your *packet name*. They are not supported within the ipkg tools.

There are more questions we do not answer here, because they are only used for source archives from the web. In this example our own sources should be in an existing directory called `/home/jbe/my-packet/` or `./home/jbe/my-packet/`. When finished this command creates a rule file called `my_packet.make` in the current directory. The new rule file doesn't know anything

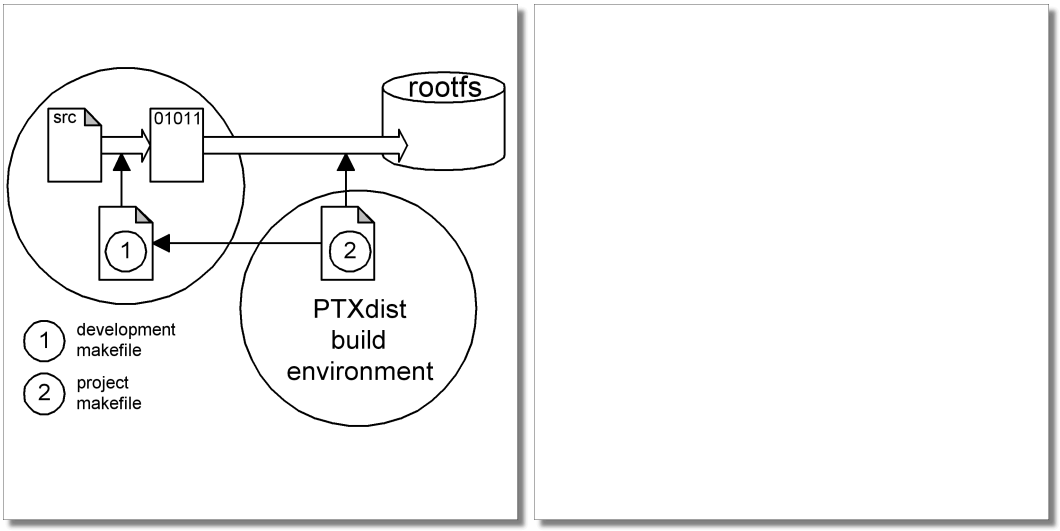

Figure 1: A rule and a make file are required to support our own sources.

specific about our development project, yet. It acts as a translator only between the development requirements and PTXdist's build mechanism, so it contains ready to use stages until `compile`. At least the `target-install` stage depends on our development requirements and must be modified manually: We must decide what files should be installed into target's root filesystem and where to install them.

To do so we open `my_packet.make` with our favorite editor and navigate to the make target called `$(STATEDIR)/my_packet.targetinstall`. The first few commands after this make target create information to be used in the ipkg mechanism.

The command we must modify starts with `@$(call install_copy`. At this point of time it is a placeholder only. It will fail later on if we use it without modification. So we modify it to a line like this:

```
@$(call install_copy, my_packet, 0, 0, 0755,
     $(MY_PACKET_DIR)/my_executable,
     /bin/my_executable)
```

This command will install the fresh built executable `my_executable` as `my_executable` into target's root filesystem in the `bin/` directory. UID and GID of the file will be both 0 (root) and the permission will be 0755. It also copies this file into a ipkg called `my_package.ipk` in the `image/` directory for later use.

For each file to be installed we insert a command line here before the line `@$(call install_finish,my_packet)`.

The last step is to activate this new rule file by extending active project's menu. Without this step the PTXdist's build environment will ignore it. See chapter **??** how to extend the menu.

Maybe there are special requirements:

- Our development project is a library that installs some binary libraries into target's root filesystem and some header files elsewhere to link other applications to this library.
- Our development project is a kernel driver that must share some structures or declarations with userspace.

To install the binary library into target's root filesystem the `target-install` mechanism shown above should be used. To let other applications include our librarie's header files, we must follow some rules to use the correct path to install them. The correct make target to do so is the `install` stage.

- If the library is intended to be used by other target packets, the binary library and its header files must be installed intalled into `$(SYSROOT)/usr/lib` and `$(SYSROOT)/usr/include`.
- If the library is a host tool or is required to build a host tool we should install our header files and libraries into `$(PTXCONF_HOST_PREFIX)/include/` and `$(PTXCONF_HOST_PREFIX)/lib/`.
- If the library is a cross tool or is required to build a cross tool we should install our header files and libraries into `$(PTXCONF_CROSS_PREFIX)/include/` and `$(PTXCONF_CROSS_PREFIX)/lib/`.

These are the default locations PTXdist's build mechanism will instruct the toolchain to search for header files and libraries we include in a `#include <friesel_frasel.h>` and `-lfrieselfrasel` manner.

Note:

- Do not install any private header file into toolchain's directory! Only the active project directory is the correct location!
- Use always the macros `SYSROOT`, `PTXCONF_HOST_PREFIX` and `PTXCONF_CROSS_PREFIX`. They all point to the active project directory (so we can build our project anywhere we like and move it around in the filesystem). If we do so, we can change the architecture later on without any modification in the makefile.
- Due to easyness of changing target's architecture (remember its only a switch in PTXdist's menu!), you **always** should write architecture independend code.

Developing

If all previous steps are successfully done, a `pxtdist go` will also build the work in progress sources and installs its results into target's root filesystem. We can test or debug the results and also rebuild in the case we fix an error. To make life easier, PTXdist supports this development cycle with a few commands:

To clean our build results:
`$ pxtdist clean my_packet`
The project makefile will call the development makefile as `make clean`.

To build our sources only:
`$ pxtdist compile my_packet`
The project makefile will call the development makefile as `make` without any argument.
If this command doesn't build anything two things may happend: PTXdist build environment "knows" that this command already run successfully. To make it "forget", run prior the command
`$ ptxdist drop my_packet compile`.
Another reason could be you didn't modify any of your sources so all targets are up do date. So it is the development makefile that decides that nothing is to do. A
`$ ptxdist clean my_packet`
should help in this case.

To install our executables into the target's root filesystem we run:
`$ pxtdist targetinstall my_packet`
The project makefile won't call the development makefile in this case, it does the installing itself.

In Case of Trouble

While building the local project ptxdist outputs this error message:

```
[...]
--------------------
target: friesel.extract
--------------------

extract: archive=/fix/path/to/source/friesel
extract: dest=/home/jbe/ptxdist-project/build-target

Unknown format, cannot extract!
```

In this case the source path was given as:

```
FRIESEL_URL  :=  file://fix/path/to/source/friesel
```

This notation uses a relative path and results into `./fix/path/to/source/friesel`, which is not what was intended. To force an absolute path, we must use three slashes:

```
FRIESEL_URL  :=  file:///fix/path/to/source/friesel
```

If our sources are part of the project, we can use the relative path notation. If we assume the sources are at `local_src/my_sources`, the path could be given as:

```
FRIESEL_URL  :=  file://local_src/my_sources
```

But it would be a good idea to use PTXdist's macros to ensure everyone can see what the author's intention was and where to find the sources:

```
FRIESEL_URL  :=  file://$(PTXDIST_WORKSPACE)/local_src/my_sources
```

Additional questions?

Below a list of locations where you can get help in case of trouble or questions how to do something special within PTXdist or general questions about Linux in the embedded world.

Mailing Lists

About PTXdist in special

This is an english language public mailing list for questions about PTXdist. See web site

http://www.pengutronix.de/mailinglists/index_en.html

how to subscribe to this list. If you want to search through the mailing list archive, visit

http://www.mail-archive.com/

and search for the list *ptxdist*.

About embedded Linux in general

This is a german language public mailing list for general questions about Linux in embedded environments. See web site

http://www.pengutronix.de/mailinglists/index_de.html

how to subscribe to this list. Note: You also can send english language mails.

News Groups

About Linux in embedded environments

This is an english language news group for general questions about Linux in embedded environments.

comp.os.linux.embedded

About general Unix/Linux questions

This is a german language news group for general questions about Unix/Linux programming.

de.comp.os.unix.programming

Chat/IRC

About PTXdist in special

irc.freenode.net:6667

Create a connection to the **irc.freenode.net:6667** server and enter the chat group **#ptxdist**. This is an english language group to answer questions about PTXdist. Best time to meet somebody in there is at europeen daytime.

Miscellaneous

Online Linux Kernel Cross Reference

A powerful cross reference to be used online.

http://lxr.linux.no/blurb.html

U-Boot manual (partially)

Manual how to survive in an embedded environment and how to use the U-Boot on target's side

http://www.denx.de/wiki/DULG

Commercial Support

You can order immediate support through customer specific mailing lists, by telephone or also on site. Ask our sales representative for a price quotation for your special requirements.

Contact us at:

Pengutronix
Hannoversche Strasse 2
D-31134 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 9

or by electronic mail:

sales@pengutronix.de

If you want to contribute to this document send your suggestions and texts under the *Creative Commons License Attribution 2.0* to *jbe@pengutronix.de*

This is a Pengutronix Application Note

**Pengutronix e.K.
Hannoversche Strasse 2
D-31134 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 9**