PTXdist Application Note

What's built?

When PTXdist builds a project it creates several subdirectories and stores bunches of files in it. This application note describes the subdirectory structure and their contents.

Pengutronix.

Directory Structure

When we take a look into a project's directory, we will find a few sub directories. After we build the project, we will find even more sub directories. Here we try to discover their meaning.

A plain Project Directory

Most of the time projects are needing patches and some kind of fixed files. More complicated projects are also needing additional rule files. For such kind of additional data, PTXdist uses specific sub directories where it expects specific data.

kernel-patches-target If target's kernel needs some patches to run on our target, PTXdist expects them here. If there is more than one patch required and the order to apply them is important, also a `series` file well be present. It contains the order from top to bottom tp apply each patch.

kernel-patches-native Same as for target's kernel. But this directory contains patches to be applied to a UML kernel for emulation purposes.

projectroot Any kind of static or specific data is located in this directory. For many menu entries we can select between a generic file from the PTXdist installation to be installes on target's root filesystem or our individual file. Whenever we select our individual file PTXdist build mechanism expects the file here.

rules If we need additional rules files to do very special things in the project, this is the location where PTXdist expects them. This is also the location if we want to replace a native rule file from the PTXdist installation. At last, new menu files are located here, if we want to extend the menu.

Documentation Some BSPs contains additional documentation. It will found here.

Beside the directories we will find some files in the main directory.

A minimalistic project directory only contains project's configuration file `ptxconfig`.

Files in the base directory and their meaning:

ptxconfig Defines everything to build in this project.

kernelconfig.target or similar Kernel configuration to be used when target's kernel is built.

Kconfig Optional. If it exists this project uses a modyfied menu.

All other directories and files are created when we are going to run the build. Here the directories first:

build-cross Contains all packages sources compiled to run on the host and handle target architecture dependend things.

build-host Contains all packages sources compiled to run on the host and handle architecture independend things.

build-target Contains all package sources compiled for the target architecure.

images Everything generated will be found here: Kernel image, rootfs image and all individual packages in ipk format.

local Contains everything target architecture dependend. A further subdirectory describes the architecture (for example i586-unknown-linux-gnu). This directory can be reached with the SYSROOT environment variable from the build system.

local-cross Contains everything that is host specific but must handle target architecture data. This directory can be accessed with PTX_PREFIX_CROSS environment variable from the build system.

local-host Contains everything that is only host specific. This directory can be accessed with PTX_PREFIX_HOST environment variable from the build system.

root Target's root filesystem image. This directory can be mounted as an NFS root for example.
Note: Due to only root can create device nodes and the build system must be run as a non root user, there is no device node present in this image. At least `/dev/console`, `/dev/null` and `/dev/zero` should exist. Create them as user root manually in order to use this directory as an NFS root.

root-debug Target's root filesystem image. The difference to `root/` is, all programs and libraries in this directory still have their debug information present. This directory is intended to be used as system root for a debugger.

state Building every package is divided onto stages. And stages of one package can depend on stages of other packages. In order to handle this correctly, this directory contains timestamp files about finished stages.

This are the generated files:

logfile Every run of PTXdist will add its output to this file. If something fails, this file can help to find the cause.

deptree.ps If the tool `dot` is installed on the host PTXdist creates a dependency tree off all packages. This file should show the complexity handled by PTXdist.

Additional questions?

Here is 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

Do you want to help improving this document?
Send your suggestions to *jbe@pengutronix.de*

This is a Pengutronix Application Note

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