OSELAS.Support
OSELAS.Training
OSELAS.Development
OSELAS.Services

Application Note
Building OSELAS.Toolchains()

Before we can start building our first userland we
need a cross toolchain. On Linux, toolchains are
no monolithic beasts. Most parts of what we need
to cross compile code for the embedded target
comes from the *GNU Compiler Collection*, gcc.
This application note describes how to build
OSELAS.Toolchains() to be used for projects
handled by PTXdist

Pengutronix.

Toolchains

Before we can start building our first userland we need a cross toolchain. On Linux, toolchains are no monolithic beasts. Most parts of what we need to cross compile code for the embedded target comes from the *GNU Compiler Collection*, gcc. The gcc packet includes the compiler frontend, gcc, plus several backend tools (cc1, g++, ld etc.) which actually perform the different stages of the compile process. gcc does not contain the assembler, so we also need the *GNU Binutils package* which provides lowlevel stuff.

Cross compilers and tools are usually named like the corresponding host tool, but with a prefix – the *GNU target*. For example, the cross compilers for ARM and powerpc may look like

- arm-softfloat-linux-gnu-gcc
- powerpc-unknown-linux-gnu-gcc

With these compiler frontends we can convert e.g. a C program into binary code for specific machines. So for example if a C program is to be compiled natively, it works like this:

```
$ gcc test.c -o test
```

To build the same binary for the ARM architecture we have to use the cross compiler instead of the native one:

```
$ arm-softfloat-linux-gnu-gcc test.c -o test
```

Also part of what we consider to be the "toolchain" is the runtime library (libc, dynamic linker). All programs running on the embedded system are linked against the libc, which also offers the interface from user space functions to the kernel.

The compiler and libc are very tightly coupled components: the second stage compiler, which is used to build normal user space code, is being built against the libc itself. For example, if the target does not contain a hardware floating point unit, but the toolchain generates floating point code, it will fail. This is also the case when the toolchain builds code for i686 CPUs, whereas the target is i586.

So in order to make things working consistently it is necessary that the runtime libc is identical with the libc the compiler was built against.

PTXdist doesn't contain a pre-built binary toolchain. Remember that it's not a distribution but a development tool. But it can be used to build a toolchain for our target. Building the toolchain usually has only to be done once. It may be a good idea to do that over night, because it may take several hours, depending on the target architecture and development host power.

Using Existing Toolchains

If a toolchain is already installed which is known to be working, the toolchain building step with PTXdist may be omitted.

The OSELAS.BoardSupport() Packages shipped for PTXdist have been tested with the OSELAS.Toolchains() built with the same PTXdist version. So if an external toolchain is being used which isn't known to be stable, a target may fail. Note that not all compiler versions and combinations work properly in a cross environment.

Every OSELAS.BoardSupport() Package checks for its OSELAS.Toolchain it's tested against, so using a different toolchain vendor requires an additional step:

Open the OSELAS.BoardSupport() Package menu with:

```
$ ptxdist platformconfig
```

and navigate to `architecture ---> toolchain` and `check for specific toolchain vendor`. Clear this entry to disable the toolchain vendor check.

Preconditions an external toolchain must meet:

- it shall be built with the configure option `--with-sysroot` pointing to its own C libraries.

- it should not support the *multilib* feature as this may confuse PTXdist which libraries are to select for the root filesystem

If we want to check if our toolchain was built with the `--with-sysroot` option, we just run this simple command:

```
$ mytoolchain-gcc -v 2>&1  grep with-sysroot
```

If this command **does not** output anything, this toolchain was not built with the `--with-sysroot` option and cannot be used with PTXdist.

Building a Toolchain

PTXdist handles toolchain building as a simple project, like all other projects, too. So we can download the OSELAS.Toolchain bundle and build the required toolchain for the OSELAS.BoardSupport() Package.

> ⚠️ Building any toolchain of the OSELAS.Toolchain-2012.12.0 is tested with PTXdist-2012.12.0. Pengutronix recommends to use this specific PTXdist to build the toolchain. So, it might be essential to install more than one PTXdist revision to build the toolchain and later on the Board Support Package if the latter one is made for a different PTXdist revision.

A PTXdist project generally allows to build into some project defined directory; all OSELAS.Toolchain projects that come with PTXdist are configured to use the standard installation paths mentioned below.

All OSELAS.Toolchain projects install their result into `/opt/OSELAS.Toolchain-2012.12.0/`.

> ⚠️ Usually the `/opt` directory is not world writeable. So in order to build our OSELAS.Toolchain into that directory we need to use a root account to change the permissions. PTXdist detects this case and asks if we want to run `sudo` to do the job for us. Alternatively we can enter:
> ```
> mkdir /opt/OSELAS.Toolchain-2012.12.0
> chown <username> /opt/OSELAS.Toolchain-2012.12.0
> chmod a+rwx /opt/OSELAS.Toolchain-2012.12.0.
> ```

We recommend to keep this installation path as PTXdist expects the toolchains at `/opt`. Whenever we go to select a platform in a project, PTXdist tries to find the right toolchain from data read from the platform configuration

settings and a toolchain at /opt that matches to these settings. But that's for our convenience only. If we decide to install the toolchains at a different location, we still can use the *toolchain* parameter to define the toolchain to be used on a per project base.

Naming Conventions Used to Describe a Toolchain

The OSELAS.Toolchain archive comes with various predefined toolchains. To select the correct one for our project we can follow the project specific documentation or, if we know what revision composition we need, we can find the correct configuration file in the following way.

For example this is one of the predefined toolchains:

i586-unknown-linux-gnu_gcc-4.7.2_glibc-2.16.0_binutils-2.22_kernel-3.6-sanitized

The first part (i586-unknown-linux-gnu) defines the target architecture and the target system. In this example it's an ia32 architecture based system, with at least i586 classes (or higher) of processors.

The next part (gcc-4.7.2) defines the compiler release, what revision of the basic C library is used (glibc-2.16.0) and what release of binutils (binutils-2.22) is used in this toolchain.

The last part (kernel-3.6) defines the kernel header in use for this toolchain. This is important, as applications built with this toolchains must be ran **at least** on this kernel revision.

We can find all predefined toolchains inside the ptxconfigs/ directory and its subdirectories.

Building a Sample OSELAS.Toolchain

To compile and install an OSELAS.Toolchain we have to extract the OSELAS.Toolchain archive, change into the new folder, configure the compiler in question and start the build.

As an example we will build the

```
i586-unknown-linux-gnu_gcc-4.7.2_glibc-2.16.0_binutils-2.22_kernel-3.6-sanitized
```

toolchain.

The steps to do so are:

> ⚠️ In order to build any of the OSELAS.Toolchains, the host must provide the tool *fakeroot*. Otherwise the message bash: fakeroot: command not found will occur and the build stops.

> ⚠️ Please ensure the 'current directory' (the . entry) is not part of your PATH environment variable. PTXdist tries to sort out this entry, but might not be successfully doing so. Check by running ptxdist print PATH if the output still contains any kind of 'current directory' as a component. If yes, remove it first.

```
$ tar xf OSELAS.Toolchain-2012.12.0.tar.bz2
$ cd OSELAS.Toolchain-2012.12.0
$ ptxdist select ptxconfigs/↵
        i586-unknown-linux-gnu_gcc-4.7.2_glibc-2.16.0_binutils-2.22_kernel-3.6-sanitized.ptxconfig
$ ptxdist go
```

At this stage we have to go to our boss and tell him that it's probably time to go home for the day. Even on reasonably fast machines the time to build an OSELAS.Toolchain is something like around 30 minutes up to a few hours.

Measured times on different machines:

- Single Pentium 2.5 GHz, 2 GiB RAM: about 2 hours

- Turion ML-34, 2 GiB RAM: about 1 hour 30 minutes

- Dual Athlon 2.1 GHz, 2 GiB RAM: about 1 hour 20 minutes

- Dual Quad-Core-Pentium 1.8 GHz, 8 GiB RAM: about 25 minutes

- 24 Xeon cores 2.54 GHz, 96 GiB RAM: about 22 minutes

When the OSELAS.Toolchain project build is finished, PTXdist is ready for prime time and we can continue with our first project.

Protecting the Toolchain

All toolchain components are built with regular user permissions. In order to avoid accidential changes in the toolchain, the files should be set to read-only permissions after the installation has finished successfully. It is also possible to set the file ownership to root. This is an important step for reliability, so it is highly recommended.

Building Additional Toolchains

The OSELAS.Toolchain-2012.12.0 bundle comes with various predefined toolchains. Refer the `ptxconfigs/` folder for other definitions. To build additional toolchains we only have to clean our current toolchain project, removing the current `selected_ptxconfig` link and creating a new one.

```
$ ptxdist clean
$ rm selected_ptxconfig
$ ptxdist select ptxconfigs/any_other_toolchain_def.ptxconfig
$ ptxdist go
```

All toolchains will be installed side by side architecture dependent into directory

`/opt/OSELAS.Toolchain-2012.12.0/architecture_part`.

Different toolchains for the same architecture will be installed side by side version dependent into directory

`/opt/OSELAS.Toolchain-2012.12.0/architecture_part/version_part`.

Additional Questions?

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

Mailing Lists

About PTXdist in Particular

This is an English language public mailing list for questions about PTXdist. See

<div align="center">

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

</div>

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

<div align="center">

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

</div>

and search for the list *ptxdist*. Please note again that this mailing list is just related to the PTXdist as a software. For questions regarding your specific BSP, see the following items.

About Embedded Linux in General

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

<div align="center">

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

</div>

on how to subscribe to this list. Note: You can also send mails in English.

About Working on the Linux Kernel

The book *Linux Kernel in a Nutshell* from *Greg Kroah-Hartman*. Its online version can be read here:

<div align="center">

http://www.kroah.com/lkn/

</div>

Chat/IRC

About PTXdist in particular

<div align="center">

irc.freenode.net:6667

</div>

Create a connection to the **irc.freenode.net:6667** server and enter the chatroom **#ptxdist**. This is an English room to answer questions about PTXdist. Best time to meet somebody there is at European daytime.

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
Peiner Str. 6-8
31137 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 55 55

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

Copyright Pengutronix e.K.
All rights reserved.

Pengutronix e.K.
Peiner Str. 6-8
31137 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 55 55

Pengutronix