OpenEmbedded

Contents

Stable: 09.12.2020 17:40 | Revision: 09.12.2020 18:38

A quality version of this page, approved on *9 December 2020*, was based off this revision.

OpenEmbbedded logo from http://www.openembedded.org

Contents

1 Overview

All the information comes from OpenEmbedded Wiki [1]

OpenEmbedded is a build framework for embedded Linux (i.e a Distribution builder). OpenEmbedded offers a cross-compile environment. It allows developers to create a complete Linux Distribution for embedded systems. Some of the OpenEmbedded advantages include:

- support for many hardware architectures
- multiple releases for those architectures
- tools for speeding up the process of recreating the base after changes have been made
- easy to customize
- runs on any Linux distribution
- cross-compiles 1000's of packages including GTK+, Qt, the X Windows system, Mono, Java, and about anything else you might ever need

One of the most famous embedded linux distibution built on top of Open Embedded is POKY (a reference distribution provided by the YOCTO project).

2 OpenEmbedded keywords

bitbake [2]: is a make-like build tool with the special focus of distributions and packages for embedded Linux cross compilation even if it's not limited to only that. It is derived from Portage, which is the package management system used by the Gentoo Linux distribution. BitBake existed for some time in the OpenEmbedded project until it was separated out into a standalone, maintained, distribution-independent tool. BitBake is co-maintained by the Yocto Project and the OpenEmbedded project.

recipes[3]: BitBake recipes specify how a particular package is built. It includes all the package dependencies, source code locations, configuration, compilation, build, install and remove instructions. It also stores the metadata for the package in standard variables. The BitBake recipes consist of the source URL (http, https, ftp, cvs, svn, git, local file system location) of the package, dependencies and compile or install options. During the build process they are used to track dependencies, performing native or cross-compile of the package and package it so that it is suitable for installation on the local or target device. OpenEmbedded has already a large list of known recipes, you may find the one you need here[4].

layers[5]: In OpenEmbedded, a layer is just a collection of recipes and/or configuration that can be used on top of OE-Core. Typically each layer is organised around a specific theme like kernel, bsp, multimedia, board... OpenEmbedded had a list of supported layers here[6].

3 OpenEmbedded way of working

The useful layers for your distribution are added in **build/conf/bblayers.conf** file.

For each recipe specified in these layers, Bitbake creates tasks. Each task represents one action to perform to cross-compile the recipe. Most of the time for one package, you have this type of tasks: fetch, configure, compile, package. Depending of the complexity of the package additional task could be done. Bitbake takes care of package dependencies and is able to perform multiple tasks at the same time.

Bitbake compiles the sources and creates distribution package (.deb or .rpm). These files could be used to populate a rootfs.

More about Layers:

- Every "meta-*" leaf directory is a layer
- A layer provides three kind of bitbake metadata with the following directory structure
  - Configuration files

```
./conf/layer.conf (layer definition)
./conf/distro/ (distribution definitions when suitable)
./conf/machine/ (machine definition when suitable)
```

  - Recipes (structure defined in layer.conf)

```
./recipes-*/ (recipe files with .bb extension)
```

  - Classes

```
./classes/ (class files with .bbclass extension)
```

Good practices with Layers:

- Only the maintainer of a layer is supposed to modify it
- Anyone willing to add or modify recipes need to create a separate layer that should be appended to the previous ones

More about recipes:

- The main goal of a linux distribution is to build deployable images or pack components for dynamic deployment
- Consequently, the most important recipes are the image recipes:
  - they require the build of all the needed output packages
  - they create the image by installing all the requested packages

4 Bitbake tips

- To create a basic rootfs:

```
bitbake core-image-minimal
```

- To get the list of recipes:

```
bitbake -s
```

- To get the list of task for one recipe:

```
bitbake recipe_name -c listtasks
```

In the list of tasks returned by the above command, remove **do_** prefix to have the name of the task.

- To force bitbake to do one specific task:

```
 bitbake recipe_name -f -c task_name
or
 bitbake recipe_name -C task_name
```

- To generate a SDK for your distribution:

```
bitbake -c populate_sdk <imagename>
```

- To recompile the kernel when config has changed:

```
bitbake virtual/kernel -f -c clean
bitbake virtual/kernel -f -c configure
bitbake virtual/kernel -f -c compile
(add -v option to see the make command used)
```

- To know the list of overlayed recipes:

```
bitbake-layers show-overlayed
```

Example:

```
bitbake-layers show-overlayed
Parsing recipes..done.%%%
=== Overlayed recipes ===
ca-certificates:
  meta-oe              20130119
  meta                 20130610
gstreamer1.0:
```

```
  meta-bsp             local
  meta                 git
  meta                 1.0.9
gstreamer1.0-plugins-bad:
  meta-bsp             local
  meta                 1.0.9
  meta                 git
gstreamer1.0-plugins-base:
  meta-bsp             local
  meta                 git
  meta                 1.0.9
...
```

5 Others useful links for OpenEmbedded

- Yocto Project Mega-Manual
- https://www.yoctoproject.org/documentation/inprogress
- http://www.openembedded.org/wiki /How_to_create_a_bitbake_recipe_for_dummies

Free Online complete training available:

- Yocto project and OpenEmbedded training

About 250 slides, with practical labs

Yocto logo from http://www.yoctoproject.org

6 Reference list

- OpenEmbedded Wiki OpenEmbedded Wiki
- bitbake
- recipes
- Layer Index,OpenEmbedded Layer Index
- layers FAQ Layers
- Layers,Supported Layers

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)