Yocto-project Documentation

Yocto-Project: - Yocto-project is an open embedded project which help to create a custom Linux environment as per our requirement or it create a Custom Linux for embedded devices. It is an open source project which is easily available all package and recipes to easily configure as our requirement to the custom Linux. Yocto project is a collection of many layers, recipes, classes and metadata.

Yocto-project uses poky to validate a yocto-project or provide distribution or environment to set the Custom linux environment.

Yocto project uses various release to use it and configure the project. In this releases have different different pacakges , configuration , recipes , layer and meta-data.

All the release have updated every 6 months like (april and October).

Release name like (kirkstone , nanbield, sumo, zeus, dunfell , gaytro many more)

This link helps to understand deeply this release:-

https://en.wikipedia.org/wiki/Yocto_Project

EOL means End of Life

In Yocto-project, Open embedded is provide a framework to yocto-project to create a basic custom linux environment. It provides various architecture supportable machine like x86-32, x86-64, qemu machine, intel machine, mircrocontroller , arm based , MIPS , Powerpc.

Yocto-project and open-embedded project share the maintainership of the main parts of the open embedded build system.

OpenEmbedded is a build automation tool and provide cross compilation framework which is used to create linux distribution for embedded devices.

OpenEmbedded build system uses bitbake tool to execute the recipes or starting the process to build the image in the host machine. It gives the instructions how to install and remove a compiled packages in my yocto-project.

Reference link - https://en.wikipedia.org/wiki/OpenEmbedded

OpenEmbedded Core :- It is mixture or collection of yocto-project and open-embedded project to set of collection a meta-data, recipes, configuration files, include files, source files, patch files, classes and many more to configure in arm based devices as well as x86 based device or provide a cross-compilation opportunity to build the image for any embedded devices or normal x86 based devices.

It provides many BSP layers like Xilinx , stm based , arm based , meta-bc , meta-ti layers

There are four major elements are important in the yocto-project :-

1. Toolchain
2. Bootloader
3. Kernel
4. Rootfile system

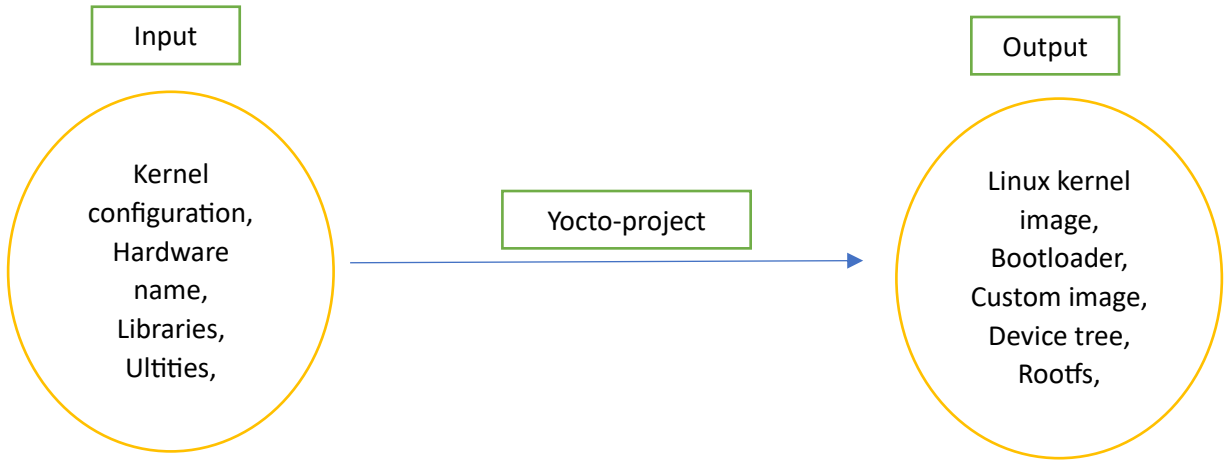
Toolchain: - Toolchain is helps to create an application or software, code to support for another device but this build in your devices. Toolchain is very important for cross compilation

Bootloader: - Bootloader initialize the program at boot time in your board and loads the kernel image in your board.

Kernel: - Kernel is the heart of the linux distribution system and it provide various functionality to manage our process, resources, memory and systemcalls.

Rootfilesystem:- Root file system define the build directories of our linux operating system and collection libraries , binary files, header files , programs and application.

Validation diagram of yocto-project :-



Important Element in Yocto-project :-

1. Poky
2. Bitbake
3. Meta
4. Build
5. Meta-poky
6. Meta-yocto-bsp
7. Meta-skeleton
8. Meta-selftest
9. Scripts
10. oe-init-build-env

Poky

Poky is a reference distribution of yocto-project. Poky helps to validate the distribution of build environment in the yocto-project. Poky have present many layers and recipes to configure the build environment.

Poky is a collection of bitbake and metadata.

Poky = bitbake + metadata

Bitbake

Bitbake is a task execution scheduler that parses the python or shell script to configure or start to build the image process.

Bitbake helps to start to configure the package and also have it remove the pacakegs.

When you run the bitbake command the main bitbake command will execute on the bitbake folder and check the binary of bitbake to start the building process.

Bitbake was originally a part of the openembedded project. It was inspired by the portage package management system used by the Gentoo linux distribution.

Bitbake is works same as make command like or is a Generic task executor.

The purpose of bitbake command is to provide :-

Single installable package means it also build the single package in your distribution.

Provide full SDK (software development kit).

Boards specific bootable linux image.

Complete with bootloader , kernel , rootfs of our build image.

Bitbake parsing the base configuration of meta-data, Recipes , Class data, configuration files and packages.

Bitbake uses two variable BBPATH and BBFILES.

BBPATH :- It is used to search for configurations and class files under the conf and classes directories.

BBFILES :- It is used to locate both recipes and recipes append files(.bb & .bbappend)

Flow of Bitbake command

do_fetch	Source-fetching
do_patch	Patching
do_unpack	Configure & Compilation
do_configure	Package Splitting
do_compile	Image Generation
do_install	SDK Generation

Every package to install in the image so this process will do bitbake to download or install or make this package part of my build image.

Source-fetching
do_fetch and do_unpack tasks fetch the source files and unpack them into the work directory.

Every patch file using SRC_URI variable this will helps to define where the source code is downloaded and save into the downloads and work directory.

TMPDIR :- is a base directory of the open embedded build system to provide the output of my build image.

do_patch :- use do_patch process tasks process recipes by using the SRC_URI variable to locate applicable patch files, which by default are (.patch or .diff) extension use .

Bitbake find the recipes and add or modify the changes in this recipes this change is save as patches form use extension (.patch or .diff).

do_configure :- this tasks configures the source by enabling and disabling any built time and configuration time options for the software being built.

do_compile :- Once a configurations task has been satisfied, Bitbake compiles the source by using the do_compile task.

do_install :- Once the compilation process is done so bitbake executes the do_install task. This task copies files from the build-directory and places then in the work directory of the build.

After the do_install process is end the use the package splitting technique. This is helps to feed the package as per user selected which types package user build the image like debian, red-hat , irist packages.
do_package_write_rpm
do_package_write_deb >> this all are the package feeds
do_package_write_ipk

“Now finally all packages are created successfully as per user select then finally build your images as per your packages “.

SCM = source control management.

SRC_URI variable uses for fetching and this point to where to fetch the source. DL_DIR variable define where to source files specifies and give instruction to open-embedded build system to create tarballs from git.

Important variable in every configuration , recipes and classes files.

SRC_URI : source variable

DL_DIR : download directory

PN : recipe name

PV = recipe version

PR = recipe revision

All the structure of bitbake files are define in bitbake.conf file

arjun@LAPTOP-EKF5QS69:~/Yocto-project/poky/meta/conf$ vim bitbake.conf

In this folder have present the bitbake file and all the bitbake command binaries are present in this folder.

arjun@LAPTOP-EKF5QS69:~/Yocto-project/poky/bitbake/bin$

this are binaries are written in python language and some time uses shell script.

Bitbake uses three things to source the configuration files by using SRC_URI variable.

Upstream Project releases :- In this project have present pre-defined small project which is used for create a small bootable image and have present some basic linux like commands for ex cat, cp , mv, poweroff many more.

Busybox , Dbus , Qt, and recipes this are the example of Upstream project releases.

Local projects :- This project have locally present in your source location to easily create a small package or basic application which is useful in linux.

For example Hello world recipe , If you write the hello world recipe then this recipes will make a part of your image and build your image then you build your final image then hello-world binaries are present in your rootfile system /usr/ bin / hello.

SCM project :- This project is specially for directly fetch the package with define your source path by using git fetcher.

For example git .

Fetcher means which thing help to fetch the source code or information.

There are many types of fetcher present here like :-

Local file fetcher

For example :- SRC_URI = file://"----------"

Http or https file fetcher

For example :- SRC_URI = https://"----------"

Git fetcher

For example :- SRC_URI = git://"----------"

Repo fetcher

For example :- SRC_URI = repo://"----------"

And many more fetcher are present here

Meta :- Meta is a collection of recipes and configuration files and classes files of machine and software. Basically meta is a layer. Layers is a collection of related recipes. In this Meta layer present have your build machine architecture configuration files and also present the bitbake configuration files and also present the default software configuration file and It have also present the SDK architecture for a specific machine.

Build :- In this folder have present the build output of my image. In this folder have present the bblayer.conf, layers.conf file which is very important to handle all the yocto-project because all packages and layers are added in this files.

Meta-poky :- In this folder have present the Distribution architecture of my yocto –project. If you set the environment then build show conf folder and in this folder have 2 main files present bblayers.conf and layers.conf file. Meta-poky helps to create this folder all the build files data are defines in this layer. Meta-poky is also a part of layer in yocto project.

Meta-yocto-bsp :- In this folder have present the predefined board support packages which is useful to build the proper image as per my board and set the proper pin configuration and supportable architecture of the field.

Oe-init-build-env :- It is a shell script file which executes the build environment to create a build directory.

Scripts :- This directory contains Various useful scripts for working with OE builds. All the scripts are present in this folder.

For example " if you run your image on qemu then qemu launch the virtual environment to give output for your image by using this script."

For example runqemu script this script mostly written in python and shell script languages.

Image specification in yocto-project :-

There are main 4 types of image are present here :-

1. Core-image-minimal

2. Core-image-full-cmdline

3. Core-image-sato

4.Core-image-western

Core-image-minimal
It is a basic command line bootable linux image. This is basic console image and provide less number of commands and configuration file. This image is basically build by using busybox projects. This image size is very small and have only low memory prints of root file system in this image.

Core-image-full-cmdline

It a full command line linux image. In this image have all functionality present same as proper linux O.S. In this image present all the editor tools and networking tools and many tools are present which is useful in the linux images. The size of this image is normal. They have present proper kernel configuration and functionalites.

Core-image-sato

It is full gui linux image like Linux O.S. In this image present all the applications as proper gui app and also present the terminal to check and run commands to configure it. This image size if larger then minimal and full-cmd line images. This is images is based on gnome graphic library and gui functionality to provide GUI based.

Core-image-western

It is aslo gui image but it present only wayland or x11 supportable terminal application not present any other gui type application. This image is basically create for show the terminal as per gui form or x11 support. In this image have present all the basic functionality same as linux image. This image file size is mostly same as sato image.