** Understand the BUILD folder in Yocto-project. **

Build folder is the main folder in yocto-project because it present all the build output of my build image.

In Build directory, its present the folder the name is **Conf**. This folder is the configuration folder of my yocto-project. **Conf** folder represent the three main files are

1. bblayers.conf
2. local.conf
3. templateconf.cfg

In build folder also present three main folder also

1. downloads
2. sstate-cache
3. Tmp

** Important Note about on BUILD **

If you set the environment by enter this command **($ source oe-init-build-env ../build/**) so build environment is set by you entering the build path and create automatically build directory ,if you not enter the path or write the build folder name is automatically create the build folder in poky directory. It is not compulsory to write the folder name is build. You can write any name of your build configuration like (**$ source oe-init-build-env ../arjun/**) so my all the build configuration data have present in the **arjun** folder name.

** This command enter in poky directory **

$ source oe-init-build-env ../arjun/
oe = open emebedded
init = initialization
build = build or create
env = environment
arjun = build configuration folder name

* If you create or set a environment then bitbake command is now working otherwise its show bitbake command not found. Bitbake is only work on openembedded build environment.

* **Source oe-init-build-env ../arjun/**: this script write in python or shell script language to help to create an open embedded build environment and it fetch the data by using **meta-poky layer**. All-important build files are present in the meta-poky layer in conf folder (**arjun@LAPTOP-EKF5QS69:~/Yocto-project/poky/meta-poky/conf$**). In this folder present all the build layers , configuration , distro and other files are here.

* Finally Build folder uses **Meta-poky** layer to initialize all the input directory like conf, temp, downloads, sstate-cache.
* This directory contains user configuration files and the output generated by the openembedded build system in its standard configurations , where the source tree is combined with the output.
* If you set or add the additional layer for your embedded board so you can changes in the **bblayers.conf file.**
* If you set or add the additional machine name, package name, architecture or sdk , or other feature for your embedded board so you can changes in the **local.conf file**.
* **Templateconf.conf** files represent the layer is used to define all the build data.

1. bblayers.conf

* bblayers.conf is a bitbake layers configuration files.
* In this files present the number of sourcing layers to fetch all the configure data during the image build process.
* The bblayers.conf file uses the bitbake layers variable to list the layers Bitbake tries to find it the source to configure the files which is present in this layers.
* Layers is a collection of related recipes which includes packages and configuration files.
* bblayers uses the poky configuration version like **2**.
* **BBPATH** :- It is used for search the configuration and class files under the conf and classes directories.
* **BBFILES** :- It is used for locate the recipes files (.bb and .bbappend).
* **BBLAYERS** :- it is used for define or declaring the layers to fetch the recipes of my image build process.

for example :-

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"
BBPATH = "${TOPDIR}"
BBFILES ?= ""
BBLAYERS ?= " \
/home/arjun/Yocto-project/poky/meta \
/home/arjun/Yocto-project/poky/meta-poky \
/home/arjun/Yocto-project/poky/meta-yocto-bsp \
 "
```

Meta , Meta-poky and Meta-yocto-bsp are the layers. In this layers have present many recipes to configure the packages. As next part we have discussed about the all the layers.

** Command to check how many layers are present in your bblayers.conf file. This command enter in Build directory folder.

$ bitbake-layers - - help

* this command show the common functionality available in bitbake-layers command and show all the related command to bitbake-layers.

for example

show-layers	=	show current configured layers.
show-overlayed	=	list overlayed recipes (where the same recipe exists in another layer)
show-recipes	=	list available recipes, showing the layer they are provided by
show-appends	=	list bbappend files and recipe files they apply to
show-cross-depends	=	Show dependencies between recipes that cross layer boundaries.
add-layer	=	Add one or more layers to bblayers.conf.
remove-layer	=	Remove one or more layers from bblayers.conf.
create-layer	=	Create a basic layer

Note :: All the layers created in poky directory , if you create a layer as per your directory so you should add correct path of your layers but your layer will be present in the same folder.

$ bitbake-layers show-layers
** this command show all the which is present in the bblayers.conf file

$ bitbake-layers add-layers (layer-name)
** this command helps to add your new creating layers in the bblayers.conf file and you can also add the path of your creating layers.

for example :-
$ bitbake-layers add-layers ../poky/my-layer

$ bitbake-layers show-recipes
** this command show all the recipes which is present in the all the layers in bblayers.conf file.

for example :-
$ bitbake-layers show-recipes git

** this command checks the recipes which is found in your yocto-project or not.
** If you check the recipes manually so you should use this command.
** for example , If you add the git package in your image then you will check first this package recipe available or not in your yocto-project. So you can enter this command and enter package name if bitbake show package name then you will directly add the package in the local.conf file (**IMAGE_INSTALL:append = " git"**).

$ bitbake-layers show-appends
** this command show all the bbappend files and recipes files which is apply in all the layers which is used in bblayers.conf.

$ bitbake-layers remove-layer (layer-name)
** this command helps to remove your layer which is found in the bblayers.conf file

$ bitbake-layers create-layer (layer-name)
** this command helps to create a new layer as per your layer name.
** First you will create the layer and second will you add this layer in bblayers.conf file.

2. templateconf.cfg

* Templateconf.cfg is a Tempalte configuration file which helps to use a meta-poky layer to determine all the Software configuration tools and files which is present in the meta-poky layer in conf folder.

* It defines the structure of your layer how to fetch the build configuration data as per use the meta-poky layers.

* Basically this file specifies only the meta-poky layer directories to directly sources up the Build process.

3. local.conf

* local.conf is a local configuration file. In this file is a collection of all machine software as well as machine hardware packages, tools, and version.

* In this file present the machine selection, Downloads source packages, shared-state-cache data, temporary image output data, distro version, package management configuration, SDK target architecture, Extra image features, image partition, and Configuration version.

* 1. **Machine Selection:-**

MACHINE ??= This variable helps to specify the machine name as per your image. There are different types of images support in the yocto-project you can directly add the machine name as per you image build.

for ex :- MACHINE ??= "qemux86-64"

qemux86-64 is a machine name which you build the image.

* 2. **Downloads:-**

DL_DIR ?= This variable helps to define all the downloads source package list of your recipes during image build process. All the data present in the form of tar files format. This downloads files are download uses a specified URL which is declare in yocto-project during the build process. In this folder present two types of files format are Tar file or normal files.

for example :-

acl-2.2.52.src.tar.gz > tar file
acl-2.2.52.src.tar.gz.done > Normal file or marker files

normal file or marker file means it verified your tar source file is download successfully by URL and ready to use this file in image build process.

DL_DIR ?= "${TOPDIR}/downloads"

** This is predefined path of your downloads source data but you can also add there own selected path.

DL_DIR ?= "/home/arjun/downloads"

** Note : First you should create downloads folder in your path then you will add this path in local.conf file. **

* 3. **Sstate-cache :-**

SSTATE_DIR ?= This variable helps to define the shared state cache of your downloads source data which is useful to optimize the build process if you build again image so it takes less time to build the new image.

SSTATE_DIR ?= "${TOPDIR}/sstate-cache"

** This is predefined path of your downloads source data but you can also add there own selected path.

SSTATE_DIR ?= "/home/arjun/sstate-cache"

** Note : First you should create sstate-cache folder in your path then you will add this path in local.conf file. **

* 4. **Tmp**:-

TMPDIR = This variable define the build output or image output data or it is a temporary folder. After the build process is complete then all the image build data are present in this folder.

* This is only for using checking the image build packages, recipes and root file system and image which is in tar form.

* If you delete this folder after image is build then don't worry if run again build process then this folder again build while it takes very low to to be ready your image output.

* As we dicussed deeply Tmp folder in next pdf version of yocto-project series.

* TMPDIR = "${TOPDIR}/tmp"

* This is predefined path of your downloads source data but you can also add there own selected path.

* TMPDIR = "/home/arjun/tmp"

* Note : First you should create tmp folder in your path then you will add this path in local.conf file.

*5. **Distro**:-

DISTRO ?= This variable specify the policy to determine which policy use yocto project to build a successfully image.

* Distro is Basically used for define a Build configuration policy.
  1. Build configuration Policy.
  2. Toolchain policy.
  3. Environment set policy.
  4. Configuration manages policy.
  5. Packages add policy.

DISTRO ?= "poky"

* poky is a distribution environment in open embedded build project which is provide the distribution to build up the Image as uses poky distribution.

* DISTRO means Distribution of Yocto project with the use of poky.

* 6. **Package Management Configuration:-**

* PACKAGE_CLASSES ?= This variable helps to understand which types of package uses to create a my images as linux format.
* This is used for create a root file system as per select you package.
* In yocto, They are three types of packages available to use to form a rootfile system as my custom linux image.

* package_deb :- for debian style deb files

* package_ipk :- for ipk files are used by opkg (a debian style embedded package manager)

* package_rpm :- for rpm style packages

rpm means *red-hat* package format uses,
deb means *debian* package format uses,
ipk means *Itsy* package format uses,

* IPK packages are a type of software package format used primarily in embedded Linux systems, and they are associated with package management systems like Opkg.

* Opkg is a lightweight package management system designed for embedded Linux systems. It is commonly used in projects like OpenWrt, which is an open-source firmware for wireless routers and other embedded devices.

PACKAGE_CLASSES ?= "package_rpm"

*7. **Extra Image Configuration defaults** :-

* EXTRA_IMAGE_FEATURES ?= This variable specify the additional image feature you want to add in your image to working as proper compatible and reliable or work as high processing and fasting.

* “dbg-pkgs" - add -dbg packages for all installed package && adds symbol information for debugging/profiling)

* "src-pkgs" - add -src packages for all installed packages && adds source code for debugging)

* "dev-pkgs" - add -dev packages for all installed packages && useful if you want to develop against libs in the image)

* "ptest-pkgs" - add -ptest packages for all ptest-enabled packages.

* "tools-sdk" - add development tools (gcc, make, pkgconfig etc.)

* "tools-debug" - add debugging tools (gdb, strace)

* "eclipse-debug" - add Eclipse remote debugging support

* "tools-profile" - add profiling tools (oprofile, lttng, valgrind)

* "tools-testapps" - add useful testing tools (ts_print, aplay, arecord etc.)

* "debug-tweaks" - make an image suitable for development && e.g. ssh root access has a blank password

EXTRA_IMAGE_FEATURES ?= "debug-tweaks"

*7. **Buildstats**:-

* USER_CLASSES ?= This Variable collecting and storing statistics about the build process.

* Its include the Build time per task and recipes.

* Memory usage log and files, CPU usage, DISK I/O operations, Network Activity.

* Its declare when was the build start, when was the build stop, when was the build successfully, When was takes time to build the image.

* If you can manually your buildstats for your image you can check by using this command,
$ bitbake –Sc core-image-minimal
or
$ bitbake –SCT core-image-minimal

USER_CLASSES ?= "buildstats"

* Note if you add any package in your image so you should always add first package in local.conf file then you will start the build your image process.

simply add this at last line in the local.conf file

IMAGE_INSTALL:append = “ git”

git package is added to my image I have checked this package on my image.

In image
$ git –version

It show the git version means your git package is successfully make a part of your image.

********** THANK – YOU **********