How to write a custom Yocto application layer for RaspberryPi

Saira Mughal · Follow
Published in Emumba · 4 min read · Jul 26, 2019

I started learning about building kernel images for any targeted SOC. On this journey, I spent my last week exploring Yocto. I found it to be a powerful tool in the Embedded world. Building kernel, compiling/adding custom

applications into Linux distribution and generating Cross-Toolchain SDK are some key features of Yocto.

One of the most significant benefits of Yocto is customization. We can add up custom layers to a Linux image for any targeted machine. Moreover, user applications can be merged into a Linux image via custom layers.

Yocto official guide has extensive documentation to add user application as a custom layer in the Yocto Project. Reading that documentation is like jumping into the sea and struggling to find your required gems. This blog is a one-click summary of that long-winded document. Developers don't need to go through the same pain that I did as I have outlined the minimum required steps to add user applications in any image. Following these steps, you will end up with a command based minimal image for RaspberryPi.

1. Cloning required repositories
2. Initializing the build environment
3. Setting the target machine
4. Creating a custom layer structure using bitbake
5. Renaming custom-layer
6. Adding metadata for your custom layer
7. Adding the custom layer in the yocto hierarchy
8. Adding application into the image
9. Generating custom image

10. Flashing image on SD card

11. Running the application

Here we go with a detailed description.

1. Cloning required repositories:

First of all clone the main Yocto project POKY

```
~$ git clone -b warrior git://git.yoctoproject.org/poky.git poky-
warrior
```

Then all the dependency layers under that:

```
~$ cd poky-warrior
~$ git clone -b warrior git://git.openembedded.org/meta-openembedded
~$ git clone -b warrior git://git.yoctoproject.org/meta-raspberrypi
```

2. Initializing the build environment

```
~$ . ./oe-init-build-env
```

The Yocto environment script will create the build directory if it does not already exist.

3. Setting the target machine

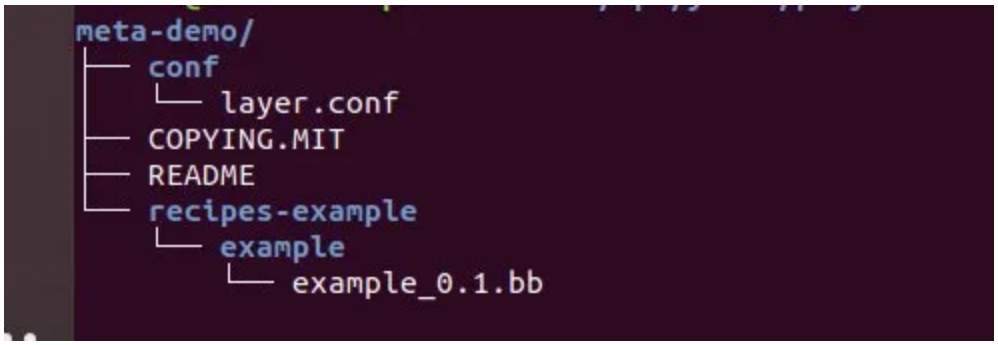
Edit **conf/local.conf** and change machine value to raspberrypi3

```
MACHINE ??= “raspberrypi3”
```

4. Creating a custom layer structure using bitbake script

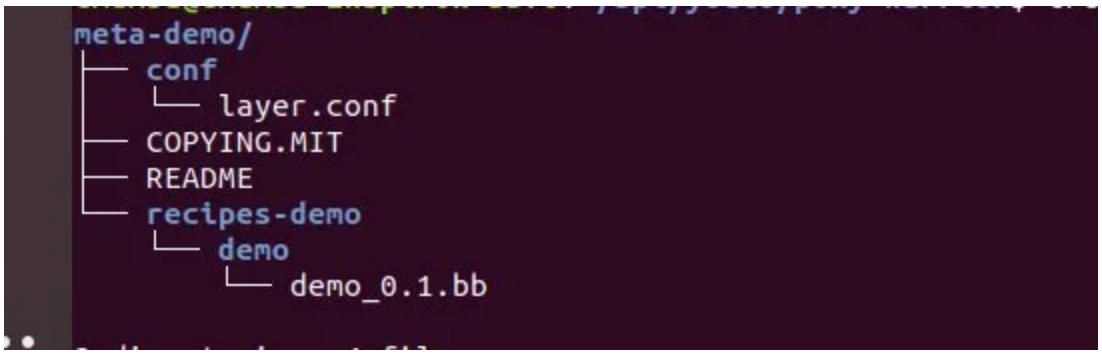
```
~$ bitbake-layers create-layer ../meta-demo
```

Bitbake script has created an example package with default values. The tree structure of meta-demo should be like this :



5. Renaming custom-layer

Replace “example” with your recipe name. For example, for the demo application you need to change as follows:



NOTE: *.bb file should be renamed as **packageName_version.bb i.e demo_0.1.bb.**

6. Adding metadata/recipes for your custom layer

Add following configurations in **meta-demo/recipes-demo/demo/demo_0.1.bb:**

```
SUMMARY = "bitbake-layers recipe"
DESCRIPTION = "Custom application"
SECTION = "examples"
LICENSE = "CLOSED"
PR = "r0"
SRC_URI =
"gitsm://github.com/waqqas/Catch2GMockDemo.git;protocol=https"
SRCREV = "c0714f912d499ad33234f5a9becb451b0170cc0b"

S = "${WORKDIR}/git"
inherit cmake
FILES_${PN} += "${bindir}"

do_install() {
    # create the /usr/bin folder in the rootfs give it default
permissions
    install -d ${D}${bindir}
    #move demo application to /usr/bin folder. in the rootfs.
 install -m 0755 demo ${D}${bindir}
}
```

We are creating a recipe that will fetch code from the given GitHub URL along with git submodule and compile code using CMake. Explanation of some related terms:

PR: Package revision or version

SRC_URI: GitHub URL and protocol. Protocols can be https, Http, git

SRCREV: Commit hash which you want to checkout

S: Source directory where Git repository will be cloned.

FILES: Files to be added in the resulting package.

Inherit: For CMake based projects we need to inherit from CMake.

NOTE: gitsm in SRC_URI is used to include submodules in the GitHub repository. If your repository does not contain any submodule you can use git.

7. Adding the newly created and dependent layers in conf/bblayer.config

```
~$ bitbake-layers add-layer ../meta-openembedded/meta-oe
~$ bitbake-layers add-layer ../meta-raspberrypi
~$ bitbake-layers add-layer ../meta-demo
```

To view newly added layer

```
~$ bitbake-layers show-layers
```

8. Making this application part of the image

Add the following statement in the build/conf/local.conf file.

```
IMAGE_INSTALL_append = “ demo”
```

9. Generating the custom image

```
~$bitbake core-image-minimal
```

It will take some time and once the build process is complete your custom application will be at:

```
~/poky-warrior/build/tmp/work/arm1176jzfshf-vfp-poky-linux-
gnueabi/demo/0.1-r0/build
```

An SD card image file with *-sdimg extension will be at

```
~/poky-warrior/build/tmp/deploy/images/raspberrypi/core-image-
minimal-raspberrypi.rpi-sdimg.
```

10. Writing image to SD card

We can flash image to sd card using dd command :

```
~$ cd ~/poky-warrior/build/tmp/deploy/images/raspberrypi
~$ dd if = core-image-minimal-raspberrypi.rpi-sdimg of=/dev/sdb bs=1
, ~$ sync
```

Insert the SD card into the RaspberryPi card slot and attach the required cables. After booting it will prompt for login id type “root” and press enter key.

11. Testing the application

The compiled application will be available at “/usr/bin/demo” to run your application:

```
~$ cd /usr/bin
~$ ./demo test
```

Today, we learned how to add a custom layer in the Yocto Project. But that’s not the end of exploring Yocto as we’ve just scratched the surface. Next, I’ll be writing about compiling Alexa SDK with AVS-SampleApp using Yocto for RaspberryPi. Fingers crossed!

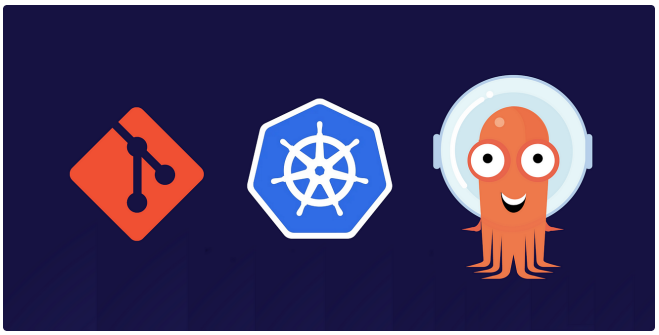
Yocto Yoctoproject Raspberry Pi

Written by Saira Mughal

5 Followers · Writer for Emumba

Follow

More from Saira Mughal and Emumba



Talha Naeem in Emumba

Mastering GitOps: Seamlessly Deploying Your App Across...

If you’re unfamiliar with GitOps, let’s demystify it. GitOps is a comprehensive framework tha...

6 min read · Oct 9

8

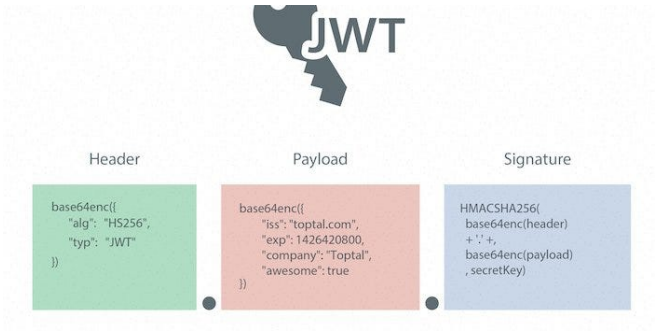
Amna Bibi in Emumba

Setup Xcode and Appium for iOS Automation (Real device)

Passing software through rounds of automation testing, before it is released into...

7 min read · Jan 24, 2022

55 1



Mah Noor in Emumba

Signing a JWT in JMeter for load testing

How to create a JWT in JMeter using JSR223 Sampler?

4 min read · Jan 6, 2022

73

Hamza Ali Imran in Emumba

How to make Python Packages part of Yocto generated Image?

Today I’m going to share with you my experience of adding Python Packages to...

5 min read · Nov 3, 2019

32 2

See all from Saira Mughal | See all from Emumba

Recommended from Medium

Aniket Fasate

“ESP-NOW: A Game Changer for ESP32 Mesh Networks and Senso...

A wireless communication protocol for quick responses and low-power control.

9 min read · Jun 15

11

Mohammad Faisal in Level Up Coding

Awesome Terminal Applications

Sharpen the axe before you cut the wood.

· 4 min read · Nov 13

20

Lists

Staff Picks

516 stories · 470 saves

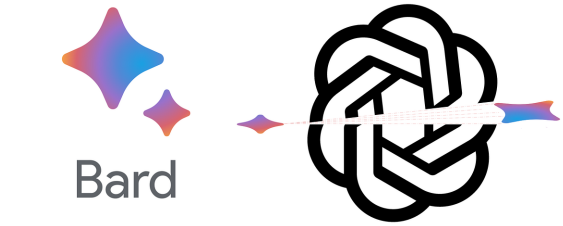
Vaishnav Manoj in DataX Journal

JSON is incredibly slow: Here's What's Faster!

Unlocking the Need for Speed: Optimizing JSON Performance for Lightning-Fast Apps...

16 min read · Sep 28

121



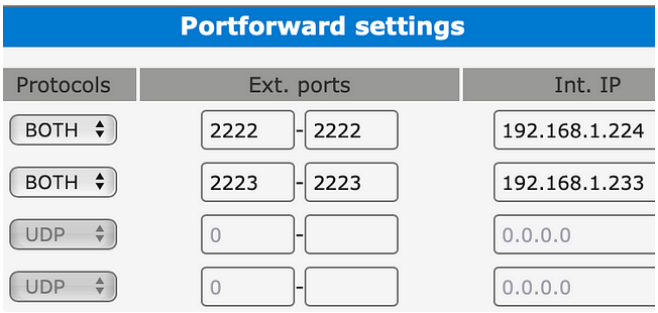
AL Anany

The ChatGPT Hype Is Over—Now Watch How Google Will Kill...

It never happens instantly. The business game is longer than you know.

· 6 min read · Sep 1

597



Paolo Molignini

How to connect to your home laptop from anywhere with SSH

I'm a researcher in quantum physics and when I tell that to non-scientists or even to other...

10 min read · May 31

Oscar Achuy

Install your own cloud storage with Nexcloud in Raspberry Pi 4

Build your Raspberry Pi cloud with a little help of Portainer

6 min read · Jun 23

1

See more recommendations