

Practice Yocto

1. Exercise 1, 2

Build image:

Step 1: Create a folder (example: yocto) and git clone poky

```
$cd yocto
```

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

```
tutorialladda@linux:~/yocto$  
tutorialladda@linux:~/yocto$ git clone git://git.yoctoproject.org/poky -b dunfell  
Cloning into 'poky'...  
remote: Enumerating objects: 523784, done.  
remote: Counting objects: 100% (523784/523784), done.  
remote: Compressing objects: 100% (123457/123457), done.  
Receiving objects: 3% (16351/523784), 7.90 MiB | 766.00 KiB/s
```

Step 2: Download Raspberry pi meta layer

```
$cd poky
```

```
$git clone https://git.yoctoproject.org/meta-raspberrypi/ -b dunfell
```

```
tutorialladda@linux:~/yocto$  
tutorialladda@linux:~/yocto$ cd poky/  
tutorialladda@linux:~/yocto/poky$  
tutorialladda@linux:~/yocto/poky$ git clone git://git.yoctoproject.org/meta-raspberrypi -b dunfell  
Cloning into 'meta-raspberrypi'...  
remote: Enumerating objects: 9169, done.  
remote: Counting objects: 100% (9169/9169), done.  
remote: Compressing objects: 100% (4502/4502), done.  
remote: Total 9169 (delta 5299), reused 7484 (delta 4229)  
Receiving objects: 100% (9169/9169), 1.82 MiB | 451.00 KiB/s, done.  
Resolving deltas: 100% (5299/5299), done.  
Checking connectivity... done.
```

Step 3: Set up an OpenEmbedded environment:

```
$source oe-init-build-env
```

```
tutorialladda@linux:~/yocto/poky$ source oe-init-build-env  
  
### Shell environment set up for builds. ###  
  
You can now run 'bitbake <target>'  
  
Common targets are:  
  core-image-minimal  
  core-image-sato  
  meta-toolchain  
  meta-ide-support  
  
You can also run generated qemu images with a command like 'runqemu qemux86'  
  
Other commonly useful commands are:  
  - 'devtool' and 'recipetool' handle common recipe tasks  
  - 'bitbake-layers' handles common layer tasks  
  - 'oe-pkgdata-util' handles common target package tasks  
tutorialladda@linux:~/yocto/poky/build$
```

Step 4: Configure the project and choose a target:

Set machine name in local.conf and add the raspberry pi layer in bblayer.conf

In file local.conf add 2 line choose machine and generate an SD card image file

For raspberry pi 3 machine name

MACHINE ?= " raspberrypi3-64"

For SD card image

IMAGE_FSTYPES = "tar.xz ext3 rpi-sdimg"

In the build/conf/bblayers.conf add path to meta-raspberrypi folder

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
/home/tutorialadda/yocto/poky/meta \
/home/tutorialadda/yocto/poky/meta-poky \
/home/tutorialadda/yocto/poky/meta-yocto-bsp \
/home/tutorialadda/yocto/poky/meta-raspberrypi \
"
```

Step 5: Start bitbake to build the image

\$bitbake core-image-minimal

```
tutorialadda@linux:~/yocto/poky/build$
tutorialadda@linux:~/yocto/poky/build$ bitbake core-image-minimal
Parsing recipes: 100% |#####|
Parsing of 805 .bb files complete (0 cached, 805 parsed). 1359 targets, 61 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.46.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-16.04"
TARGET_SYS           = "aarch64-poky-linux"
MACHINE              = "raspberrypi4-64"
DISTRO               = "poky"
DISTRO_VERSION        = "3.1.10"
TUNE_FEATURES        = "aarch64 cortexa72 crc crypto"
TARGET_FPU           = ""
meta
meta-poky
meta-yocto-bsp       = "dunfell:7d2f118cb6f5e140dbde794200b7c8cb07ae1b10"
meta-raspberrypi     = "dunfell:59c2d6f7a8b1239bd7b587b9180c2a55f9c695a2"

NOTE: Fetching uninitative binary shin http://downloads.yoctoproject.org/releases/uninitative/3.2/x86_64-nativesdk-libc.tar
2d4c7ae3887cddb97219f97b94efddfeee2e24923c0cb0e8ce84c6 (will check PREMIRRORS first)
Initialising tasks: 100% |#####|
Sstate summary: Wanted 1158 Found 0 Missed 1158 Current 0 (0% match, 0% complete)
NOTE: Executing Tasks
Currently 2 running tasks (13 of 3150)  0% |
0: m4-native-1.4.18-r0 do_fetch (pid 20832) 66% |#####|
1: gnu-config-native-20200117+gitAUTOINC+5256817ace-r0 do_fetch (pid 20830) 43% |#####|
```

Boot to board with TFTP:

After build success, you need to 3 files: device tree, rootfs and Image***.bin in
poky/build/tmp/deploy/images/raspberrypi4-64/

```
bcm2710-rpi-3-b-plus-1-5.4.72+gitAUTOINC+5d52d9eea9_154de7bbd5-r0-raspberrypi3-64-20231208043512.dtb
bcm2710-rpi-3-b-plus.dtb
bcm2710-rpi-3-b-plus-raspberrypi3-64.dtb
bcm2710-rpi-3-b-raspberrypi3-64.dtb
```

```
i2c-rtc-raspberrypi3-64.dtbo
Image
Image-1-5.4.72+gitAUTOINC+5d52d9eea9_154de7bbd5-r0-raspberrypi3-64-20231208043512.bin
Image-raspberrypi3-64.bin
iqaudio-dac-1-5.4.72+gitAUTOINC+5d52d9eea9_154de7bbd5-r0-raspberrypi3-64-20231208043512.dtbo
iqaudio-dac.dtbo
```

```
core-image-minimal-raspberrypi3-64-20231208043512.rootfs.manifest
core-image-minimal-raspberrypi3-64-20231208043512.rootfs.tar.bz2
core-image-minimal-raspberrypi3-64-20231208043512.rootfs.wic.bmap
core-image-minimal-raspberrypi3-64-20231208043512.rootfs.wic.bz2
core-image-minimal-raspberrypi3-64-20231208043512.testdata.json
```

Step 1: Rename file *.bin to “Image”, move it and file device tree to:

/home/tftpboot/rpi3/(board_target)/(your_folder)

Example: /home/tftpboot/rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/

Step 2: Extract file rootfs to:

/home/nfs/rpi3/(board_target)/(your_folder)

Example: /home/nfs/rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/

```
hao.tran-quang@soclab01:/home/nfs/rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang$ ls
bin boot dev etc home lib media mnt proc run sbin sys tmp usr var
```

Step 3: Config file in /home/tftpboot/pxelinux.cfg/(MAC's board)

Example: 01-b8-27-eb-01-e2-f8

```
hao.tran-quang@soclab01:/home/tftpboot/pxelinux.cfg$ ls
01-48-b0-2d-2e-5d-3a 01-b8-27-eb-01-e2-f8 01-b8-27-eb-4b-a1-91 01-b8-27-eb-c5-ec-6d
01-48-b0-2d-2e-5d-3a_jetson 01-b8-27-eb-01-e2-f8_rpi 01-b8-27-eb-4b-a1-91_rpi 01-b8-27-eb-c5-ec-6d_rpi
hao.tran-quang@soclab01:/home/tftpboot/pxelinux.cfg$ vi 01-b8-27-eb-01-e2-f8
```

Add your label:

LABEL your_label

MENU your_menu

LINUX rpi3/(board_target) /(your_folder)/Image

FDT rpi3/(board_target) /(your_folder)/bcm2710-rpi-3-b-plus.dtb

APPEND \${cbootargs} \${bootargs} 8250.nr_uarts=1 console=ttyS0,115200 rw ip=dhcp
root=/dev/nfs

nfsroot=192.168.200.100:/home/nfs/rpi3/(board_target)/(your_folder)/,nfsvers=3,tcp

Example:

LABEL quanghao

MENU quanghao

LINUX rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/Image

FDT rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/bcm2710-rpi-3-b-plus.dtb

APPEND \${cbootargs} \${bootargs} 8250.nr_uarts=1 console=ttyS0,115200 rw ip=dhcp
root=/dev/nfs nfsroot=192.168.200.100:/home/nfs/rpi3/01-b8-27-eb-01-e2-f8/hao.tran-
quang,nfsvers=3,tcp

Step 4: Access to board you want to boot by minicon command:

Raspberry Pi (MAC: B8:27:EB:4B:A1:91)

UART debug(../ttyUSB0): minicom -D /dev/serial/by-path/pci-0000\000\1a.0-usb-
0\1.4.1.4\1.0-port0

UART debug (ttyUSB1): minicom -D /dev/serial/by-path/pci-0000\000\1a.0-usb-
0\1.4.1.3\1.0-port0

Step 5: Boot OS:

```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/serial/by-path/pci-0000:00:1a.0-usb-0:1.4.1.3:1.0-port0, 11:52:29

Press CTRL-A Z for help on special keys

root@raspberrypi3-64:~# reboot
```

```

549.8 KiB/s
done
Bytes transferred = 6766 (1a6e hex)
Config file '<NULL>' found
Ignoring malformed menu command: bac
Ignoring malformed menu command: cong
Ignoring malformed menu command: phuc
Ignoring malformed menu command: trai
Ignoring malformed menu command: tri
Ignoring malformed menu command: thuan
Ignoring malformed menu command: quan2
Ignoring malformed menu command: nghia
Ignoring malformed menu command: nghia
Ignoring malformed menu command: anh
Ignoring malformed menu command: device
Ignoring malformed menu command: device
Ignoring malformed menu command: quocpham
Ignoring malformed menu command: quanghao
PXELinux boot options
1:      default kernel on TFTP - DO NOT EDIT
2:      template develop lable
3:      bachuynh
4:      danh
5:      phucle
6:      trai
7:      tri
8:      thuan
9:      quan2
10:     nghia
11:     nghia2
12:     nghia2
13:     thuan.nguyen-hong
14:     anhdlt
15:     minh.nguyen-hoang2
16:     maohuynh
17:     quocpham
18:     quanghao
Enter choice: 18

```

Choose your label

```

Retrieving file: rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/Image
lan78xx_eth Waiting for PHY auto negotiation to complete..... done
Using lan78xx_eth device
TFTP from server 192.168.200.100; our IP address is 192.168.200.199
Filename 'rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/Image'
Load address: 0x800000
Loading: ##### 16.9 MiB
3.1 MiB/s
done
Bytes transferred = 17728001 (10e8201 hex)
append: root=/dev/nfs ip=192.168.200.199:::eth0 nfsroot=192.168.200.100:/home/nfs/rpi3/01-b8-27-eb-01-e2-f8/phuc,nfsvers=3 rw 8250.nr_uploads=1 console=ttyS0,115200 rw ip=dhcp root=/dev
Retrieving file: rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/bcm2710-rpi-3-b-plus.dtb
lan78xx_eth Waiting for PHY auto negotiation to complete..... done
Using lan78xx_eth device
TFTP from server 192.168.200.100; our IP address is 192.168.200.199
Filename 'rpi3/01-b8-27-eb-01-e2-f8/hao.tran-quang/bcm2710-rpi-3-b-plus.dtb'
Load address: 0x2600000
Loading: ##### 27.9 KiB
557.6 KiB/s
done
Bytes transferred = 28599 (6fb7 hex)
## Flattened Device Tree blob at 02600000
Booting using the fdt blob at 0x2600000
Using Device Tree in place at 0000000002600000, end 0000000002609fb6

Starting kernel ...
[ 0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[ 0.000000] Linux version 5.4.72-v8 (oe-user@oe-host) (gcc version 9.5.0 (GCC)) #1 SMP PREEMPT Mon Oct 19 11:12:20 UTC 2020
[ 0.000000] Machine model: Raspberry Pi 3 Model B+
[ 0.000000] efi: Getting EFI parameters from FDT:
[ 0.000000] efi: UEFI not found.
[ 0.000000] Reserved memory: created CMA memory pool at 0x0000000037400000, size 64 MiB
[ 0.000000] OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool

```

Device Tree and Image loaded

```

INIT: version 2.99 booting
Starting udev
[ 11.007430] udevd[122]: starting version 3.2.10
[ 11.267738] udevd[123]: starting eudev-3.2.10
Sun Mar 25 23:41:51 UTC 2018
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
ifup skipped for nfsroot interface eth0
run-parts: /etc/network/if-pre-up.d/nfsroot: exit status 1
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 3.4.1 raspberrypi3-64 /dev/ttyS0
raspberrypi3-64 login: root

```

```

root@raspberrypi3-64:~# uname -r
5.4.72-v8

```

2. Exercise 3, 4, 5

Step 1: Create a new meta layer and add it to bblayer.conf file

\$source oe-init-build-env

```

hao.tran-quang@soclab01:~/poky$ source oe-init-build-env

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-sato
  meta-toolchain
  meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemux86'

Other commonly useful commands are:
- 'devtool' and 'recipetool' handle common recipe tasks
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks

```

\$bitbake-layers create-layer ../meta-custom

```

hao.tran-quang@soclab01:~/poky/build$ bitbake-layers create-layer ../meta-custom
NOTE: Starting bitbake server...
NOTE: Bitbake server didn't start within 5 seconds, waiting for 90
Add your new layer with 'bitbake-layers add-layer ../meta-custom'
hao.tran-quang@soclab01:~/poky/build$

```

\$bitbake-layers add-layer ../meta-custom

```

hao.tran-quang@soclab01:~/poky/build$ bitbake-layers add-layer ../meta-custom
NOTE: Starting bitbake server...
hao.tran-quang@soclab01:~/poky/build$

```

bitbake-layers show-layers

```
hao.tran-quang@soclab01:~/poky$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                path                                     priority
=====
meta                  /home/hao.tran-quang/poky/meta           5
meta-poky             /home/hao.tran-quang/poky/meta-poky      5
meta-yocto-bsp        /home/hao.tran-quang/poky/meta-yocto-bsp 5
meta-raspberrypi      /home/hao.tran-quang/poky/meta-raspberrypi 9
meta-custom           /home/hao.tran-quang/poky/meta-custom     6
```

```
hao.tran-quang@soclab01:~/poky/meta-custom$ tree
.
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
├── recipes-example
│   └── example
│       └── example_0.1.bb
```

Step 2: Create Directory For Recipe and Source Files

Our meta custom layer directory structure looks like this.

```
hao.tran-quang@soclab01:~/poky/meta-custom$ tree
.
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
├── recipes-example
│   ├── example
│   │   └── example_0.1.bb
│   └── hello
│       ├── files
│       │   └── hello.c
│       └── hello_1.0.bb
```

We need to create a hello and files directory at the below location.

poky/meta-tutorial/recipe-example/hello

poky/meta-tutorial/recipe-example/hello/files/

Step 3: Write the simple hello world C program

Create the hello.c file at the poky/meta-tutorial/recipe-example/hello/files/hello.c

```
//Simple Hello World Program
```

```
#include<stdio.h>
```

```
int main() {
```

```
printf("Hello World , Created Bitbake recipe successfully\n");
```



```
return 0;
```

```
}
```

Step 4: Write the simple hello recipe file

Create hello_1.0.bb recipe file at the **poky/meta-tutorial/recipe-example/hello/hello_1.0.bb**

```
DESCRIPTION = "Simple helloworld application"
```

```
LICENSE = "MIT"
```

```
LIC_FILES_CHKSUM =
```

```
"file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"
```

```
SRC_URI = "file://hello.c"
```

```
S = "${WORKDIR}"
```

```
do_compile() {
```

```
    ${CC} hello.c ${LDFLAGS} -o hello
```

```
}
```

```
do_install() {
```

```
    install -d ${D}${bindir}
```

```
    install -m 0755 hello ${D}${bindir}
```

```
}
```

- This hello recipe fetch the source file(hello.c) using the SRC_URI variable and do_compile used to compile the hello.c source file and generated the hello binary.
- do_install function install hello binary at the /usr/bin of the target rootfs.
- Now, the latest directory looks like this.

```
hao.tran-quang@soclab01:~/poky/meta-custom$ tree
.
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-example
    ├── example
    │   └── example_0.1.bb
    └── hello
        ├── files
        │   └── hello.c
        └── hello_1.0.bb

5 directories, 6 files
```


Step 5: Add hello package to rootfs

Add in conf/local.conf file:

```
IMAGE_INSTALL_append = "hello"
```

Step 6: Build Image

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

```
hao.tran-quang@soclab01:~/poky$ bitbake core-image-minimal
Loading cache: 100% |#####| Time: 0:00:00
Loaded 1364 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION      = "1.46.0"
BUILD_SYS       = "x86_64-linux"
NATIVELSBSTRING = "universal"
TARGET_SYS      = "aarch64-poky-linux"
MACHINE         = "raspberrypi3-64"
DISTRO          = "poky"
DISTRO_VERSION  = "3.1.29"
TUNE_FEATURES   = "aarch64 cortexa53 crc"
TARGET_FPU      = ""
meta
meta-poky
meta-yocto-bsp   = "dunfell:b8f1972b8482860d649641ad34ec8a17ef1dd983"
meta-raspberrypi = "dunfell:2081e1bb9a44025db7297bfd5d024977d42191ed"
meta-custom     = "dunfell:b8f1972b8482860d649641ad34ec8a17ef1dd983"

Initialising tasks: 100% |#####| Time: 0:00:46
Sstate summary: Wanted 10 Found 0 Missed 10 current 1157 (0% match, 99% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3168 tasks of which 3142 didn't need to be rerun and all succeeded.
```

Step 7: Boot Image (Same in Exercise 1)

```
Poky (Yocto Project Reference Distro) 3.1.29 raspberrypi3-64 /dev/ttyS0
raspberrypi3-64 login: root
root@raspberrypi3-64:~# hello
Hello World , Created Bitbake recipe successfully
root@raspberrypi3-64:~#
```

3. Exercise 6:

Step 1: Create file “your_machine.bb” in /poky/meta/recipes-core/images

Example: core_image_bv.bb

```
hao.tran-quang@soclab01:~/poky/meta/recipes-core/images$ ls
build-appliance-image  core-image-base.bb  core-image-minimal.bb  core-image-minimal-initramfs.bb  core-image-tiny-initramfs.bb
build-appliance-image_15.0.0.bb  core-image-bv.bb  core-image-minimal-dev.bb  core-image-minimal-mtdutils.bb
```

Step 2: Open file and add line “require recipes-core/images/core-image-minimal.bb” to have a machine with name is “your_machine” and it inherit core-image-minmal, you can add some package or more install something.

Step 3: Build image:

```
$bitbake core_image_bv.bb
```

```
core-image-bv-raspberrypi3-64-20231214074157.rootfs.ext3
core-image-bv-raspberrypi3-64-20231214074157.rootfs.manifest
core-image-bv-raspberrypi3-64-20231214074157.rootfs.rpi-sdimg
core-image-bv-raspberrypi3-64-20231214074157.rootfs.tar.xz
core-image-bv-raspberrypi3-64-20231214074157.testdata.json
core-image-bv-raspberrypi3-64.ext3
core-image-bv-raspberrypi3-64.manifest
core-image-bv-raspberrypi3-64.rpi-sdimg
core-image-bv-raspberrypi3-64.tar.xz
core-image-bv-raspberrypi3-64.testdata.json
```

Exercise 7:

```
root@raspberrypi3-64:~# vim test.txt
```

A screenshot of a Kali Linux desktop environment. The background is a dark, textured wallpaper. On the left side, there is a vertical dock containing icons for a web browser, a terminal, and a file manager. The top of the screen features a taskbar with several open application windows. The bottom of the screen displays a system status bar with various indicators and system information.

```
root@raspberrypi3-64:~# ls
test.txt
root@raspberrypi3-64:~#
```