Introduction to the Yocto Project

Developer’s perspective
Igor Baldachini

Tiziano Cappellari

http://www.nabertech.com/
What is the Yocto Project

- An open source project hosted at The Linux Foundation
- A collection of:
  - embedded projects
  - application development tools
Goals

● Help to build a Linux distribution for embedded systems
● Improving the software development process for embedded Linux distributions

“Is not an Embedded Linux Distribution - It creates a custom one for You!”
Benefits

- Support for all major embedded archs
  - x86, x86-64, ARM, PPC, MIPS
- Start with a validated collection of software
- Access to a collection of application developer tools
- Modular development through the Layer model
Build System: Poky

Poky consists of:

- **Bitbake**: execute and manage all the build steps
- **Metadata**: task definitions:
  - Configuration (.conf): global definition of variables
  - Classes (.bbclass): define the build logic, the packaging ...
  - Recipes (.bb): defines the individual piece of software/image to be build
Recipes

Contains the following metadata:

- Repository or Path of the source code of the packages to build
- Patches to apply
- Dependencies from other recipes or from libraries
- Configuration and compilation options
- Define the packages to create and what files goes into the packages.
Recipes Build Process

Bitbake build a recipe following this steps:

- **fetch and unpack:**
  - can get the source files from tarballs, git, svn, etc.
  - the source files are extracted into the work directory
  - and packed into download directory for future builds.

- **patch:**
  - the extracted source files are then patched
Recipes Build Process (cont)

- configure and install
  - many standard build rules are available as autotools, cmake, gettext
  - put the build into the staging area
- package generation
  - create packages for dev, docs, locales
  - support the formats ipk, Debian, RPM
Layers

- OpenEmbedded Core Metadata (meta)
- Yocto specific layer metadata (meta-yocto)
- Hardware Specific BSP
- UI specific Layer
- Commercial Layer
- Developer specific Layer
- OpenEmbedded Core Metadata (meta)
Quick Start
Core image for qemu-x86

Get the Yocto Project:

https://www.yoctoproject.org/downloads

or run:

```
git clone -b daisy git://git.yoctoproject.org/poky.git
```

Init the environment

```
source oe-init-build-env
```
Core image for qemux86 (cont)

Create the image
bitbake core-image-minimal

Run emulator
runqemu qemux86
Poky Folders
Overview
Configuration files for the build environment

downloaded upstream source tarballs

deployed images, sdk and packages

shared header files and libraries for share files between packages

where all packages has its own directory and where bitbake unpack, patch, configure and compile

deploy sysroots

work

tmp

downloads

conf

poky

build

bitbake
documentation
meta
meta-yocto
meta-yocto-bsp
meta-self-test
meta-skeleton
scripts
poky
  build
  bitbake ➔ bitbake executable
  documentation ➔ Yocto project documentation
  meta
  meta-yocto
  meta-yocto-bsp
  meta-self-test
  meta-skeleton
  scripts ➔ Yocto scripts for extra functionalities (like qemu, hob, ...)
Contains the OpenEmbedded Core metadata.

Core set of configuration files

Core recipes

Contains the *.bbclass files that are used to abstract common code so it can be reused by multiple packages
poky

build

documentation

meta

meta-yocto → Configuration for the poky reference distribution

meta-yocto-bsp → Yocto reference hardware BSP

meta-selftest → Used to verify the behavior of the build system

meta-skeleton → Template recipes for BSP and kernel development

scripts
Project Example
Monitor
Specifications

- Read the status of 8 switches.
- Active 8 Relays accordingly with the switches status
- Show the switches status on a display
Hardware

- **SBC**: Single Board Computer
- **DiDo**: Device that get the status from 8 opto-isolated inputs and set the status to 8 relays. The communication with the host is made through an RS232 serial line
- **Display**: HDMI display
Software

- **Monitor**: application that display the status of switches, based on the GTK+ widget toolkit
- **DiDo Library**: handles the communication with a DiDo device
Test Monitor with qemu
Create a new Layer: meta-monitor

- Create poky subfolder: meta-monitor
- add the new layer into build/conf/bblayers.conf

```bash
# [...] 
BBLAYERS ?= " \ 
 /poky-absolute-path/poky/meta \ 
 /poky-absolute-path/poky/meta-yocto \ 
 /poky-absolute-path/poky/meta-yocto-bsp \ 
 /poky-absolute-path/poky/meta-monitor \ 
 " 
# [...] 
```
Create DiDo library recipe

Add the new recipe file:

```
meta-monitor/recipes-monitor/dido/dido_1.0.0.bb
```

```
SUMMARY="Protocol Communication Library for the DiDo devices"
HOMEPAGE="www.example.com/dido"
DEPENDS="boost"
LICENSE="GPLv2+"
LIC_FILES_CHKSUM="file://COPYING;md5=..."
SRCREV="..."
SRC_URI="git://com.example/sw/libdido;protocol=ssh"
S="${WORKDIR}/git"
inherit cmake pkgconfig
```
Create DiDo library recipe

Add the new recipe file:

```
meta-monitor/recipes-monitor/dido/dido_1.0.0.bb
```

SUMMARY="Protocol Communication Library for the DiDo devices"
HOMEPAGE="www.example.com/dido"
DEPENDS="boost"
LICENSE="GPLv2+
LIC_FILES_CHKSUM="file://COPYING;md5=..."
SRCREV="...
SRC_URI="git://com.example/sw/libdido;protocol=ssh"
S="${WORKDIR}/git"
inherit cmake pkgconfig"
Create DiDo library recipe

Add the new recipe file:

```makefile
meta-monitor/recipes-monitor/dido/dido_1.0.0.bb
```

```
SUMMARY="Protocol Communication Library for the DiDo devices"
HOMEPAGE="www.example.com/dido"
DEPENDS="boost"
LICENSE="GPLv2+"
LIC_FILES_CHKSUM="file://COPYING;md5=...
SRCREV="...
SRC_URI="git://com.example/sw/libdido;protocol=ssh"
S="${WORKDIR}/git"
inherit cmake pkgconfig
```

The revision of the source code used to build the package

How to fetch the source files
Create DiDo library recipe

Add the new recipe file:
meta-monitor/recipes-monitor/dido/dido_1.0.0.bb

SUMMARY="Protocol Communication Library for the DiDo devices"
HOMEPAGE="www.example.com/dido"
DEPENDS="boost"
LICENSE="GPLv2+
LIC_FILES_CHKSUM="file://COPYING;md5=..."
SRCREV="..."
SRC_URI="git://com.example/sw/libdido;protocol=ssh"
S="${WORKDIR}/git"
inherit cmake pkgconfig

WORKDIR:
tmp/work/qemux86-poky-linux/dido/1.0.0-r0

The location in the Build Directory where unpacked recipe source code resides
Create DiDo library recipe

Add the new recipe file:
meta-monitor/recipes-monitor/dido/dido_1.0.0.bb

SUMMARY="Protocol Communication Library for the DiDo devices"
HOMEPAGE="www.example.com/dido"
DEPENDS="boost"
LICENSE="GPLv2+"
LIC_FILES_CHKSUM="file://COPYING;md5="
SRCREV="..."
SRC_URI="git://com.example/sw/libdido;protocol=ssh"
S="${WORKDIR}/git"

Causes the named class to be inherited at this point during parsing.
Create Monitor library recipe

Add the new recipe file:

```
meta-monitor/recipes-monitor/monitor/monitor_1.0.0.bb
```

```sh
SUMMARY="Monitorize DiDo device"
HOMEPAGE="www.example.com/monitor"
DEPENDS="dido gtk+"
LICENSE="GPLv2+"
LIC_FILES_CHKSUM="file://COPYING;md5=..."
SRCREV="...
SRC_URI="git://com.example/sw/monitor;protocol=ssh"
S="${WORKDIR}/git"
```

# continue...
Create Monitor library recipe (cont)

meta-monitor/recipes-monitor/monitor/monitor/monitor_1.0.0.bb

FILES_${PN}+=${sysconfdir}/init.d/monitor-boot

INITSCRIPT_PARAMS = "defaults 99"

INITSCRIPT_NAME = "monitor-boot"

inherit cmake pkgconfig update-rc.d

List of directories or files that are placed in the package

The filename of the initialization script as installed to ${sysconfdir}/init.d

Specifies the name of the script and the options to pass to update-rc.d

inherit update-rc.d.bbclass
Create Monitor Image recipe

Add the new recipe file:
meta-monitor/recipes-core/images/monitor-image.bb

IMAGE_FEATURES += "package-management x11-base"
inherit core-image

IMAGE_INSTALL += "dido monitor"

Features to include in an image as defined into image.bbclass file.

inherit from core-image.bbclass

Specifies the packages to install into the image
Create Monitor layer configuration

Add the new configuration file:

`meta-monitor/conf/layer.conf`

- **BBPATH**
  
  ```
  ::= "${LAYERDIR}"
  ```

- **BBFILES**
  
  ```
  += "${LAYERDIR}/recipes-*/*/*.bb"
  ```

- **BBFILE_COLLECTIONS**
  
  ```
  += "monitorsystem"
  ```

- **BBFILE_PATTERN_monitorsystem**
  
  ```
  := "^${LAYERDIR}/"
  ```

- **BBFILE_PRIORITY_monitorsystem**
  
  ```
  = "5"
  ```

Add to BBPATH the current layer directory. Used by bitbake to locate . bbclasses and configuration files.

Append all recipes.

Append the layer name. These names are used to find the other BBFILE_* variables.
Create Monitor layer configuration

Add the new configuration file:
meta-monitor/conf/layer.conf

BBPATH .= "${LAYERDIR}"

BBFILES += "${LAYERDIR}/recipes-*/**/*.bb "

BBFILE_COLLECTIONS += "monitorsystem"
BBFILE_PATTERNS_monitorsystem := "^${LAYERDIR}/"
BBFILE_PRIORITY_monitorsystem = "5"

Set to a regular expression. Is used to match files from BBFILES into a particular layer.

Assigns a priority to the layer. Allows to choose the layer that takes precedence when the same recipe appear in multiple layers.
Test Monitor with qemu

Create the image

```
bitbake monitor-image
```

Run emulator

```
runqemu qemu-x86 monitor-image qemuparams="-usb -usbdevice serial::/dev/ttyUSB0"
```
Monitor on BeagleBone
Hardware

Processor: AM335x 720MHz ARM Cortex-A8
- 256MB DDR2 RAM
- 3D graphics accelerator
- ARM Cortex-M3 for power management
- 2x PRU 32-bit RISC CPUs

Connectivity
- USB client: power, debug and device
- USB host
- Ethernet
- 2x 46 pin headers
Configure the new machine

Beagle Bone BSP is already contained on meta-yocto-bsp layer:

```
meta-yocto-bsp/conf/machine/beaglebone.conf
```

Configure the new machine on:

```
build/conf/local.conf
```

```
# Machine Selection
MACHINE != "beaglebone"
#...
```
Build the image and Deploy (uSD)

Build

```
bitbake monitor-image
```

Create this partitions on uSD:

- FAT32 partition with boot flag, 30MB is enough
- EXT4 partition

```
cp build/tmp/deploy/images/beaglebone/MLO-beaglebone
      path_of_the_first_partition/MLO

cp build/tmp/deploy/images/beaglebone/u-boot-beaglebone.img
      path_of_the_first_partition/u-boot.img

tar -xvf build/tmp/deploy/images/beaglebone/monitor-image.tar.bz2
     -C path_of_the_second_partition
```
Monitor on Seco QuadMo747-X/i.MX6
QuadMo747-X/i.MX6

Freescale™ i.MX6, based on ARM Cortex-A9 processors:

- 4GB DDR3 onboard
- 2D, OpenGL® ES2.0 3D and OpenVG™ accelerators

Connectivity:

- HDMI interface
- eMMC soldered onboard
- MMC/SD/SDIO interface
- 1 x μSD card slot onboard
- Gigabit Ethernet interface
- AC97 Audio Interface
- 1x USB OTG
- 4x USB 2.0 Host
BSP

On [https://www.yoctoproject.org/downloads/bsp](https://www.yoctoproject.org/downloads/bsp) the QuadMo747-X/i.MX6 is missing.

All that we have is the Kernel Linux from the Seco ufficial site.
Create a custom BSP

Example filesystem layout:

```
meta-bsp_name/
  bsp_license_file
  README
  README.sources
  conf/layer.conf
  conf/machine/
  recipes-core/
  recipes-graphics/
  recipes-kernel/linux/
```

- **meta-bsp_name/**: Special information needed for building the image.
- **bsp_license_file**: Provides information on where to locate the BSP source files.
- **README**: Hardware configuration options.
- **README.sources**: Miscellaneous and graphics BSP-specific recipes.
- **conf/layer.conf**: Linux kernel configuration.
- **conf/machine/**: Hardware configuration options.
- **recipes-core/**: Miscellaneous and graphics BSP-specific recipes.
- **recipes-graphics/**: Miscellaneous and graphics BSP-specific recipes.
- **recipes-kernel/linux/**: Linux kernel configuration.
Configure the build

Add the new BSP Layer on

`build/conf/bblayers.conf`

```
# [...]  
BBLAYERS ?= " 
   /poky-absolute-path/poky/meta 
   /poky-absolute-path/poky/meta-yocto 
   /poky-absolute-path/poky/meta-yocto-bsp 
   /poky-absolute-path/poky/meta-seco 
   /poky-absolute-path/poky/meta-monitor 
   "  
# [...]`
```
Configure the build (cont)

Set the new machine on:

build/conf/local.conf

# [...]# Machine Selection
MACHINE ?= "quadmo747x1Mx6"
# [...]
Create BSP layer configuration

Add the new configuration file:

`meta-seco/conf/layer.conf`

```bash
BBPATH := "${LAYERDIR}"

BBFILES += "${LAYERDIR}/recipes-*/*/*.bb"

BBFILE_COLLECTIONS += "seco"
BBFILE_PATTERN_seco := "^${LAYERDIR}/"
BBFILE_PRIORITY_seco = "5"
```
Create New Machine

Add the new machine configuration:
meta-seco/conf/machine/quadmo747xiMx6.conf

```plaintext
#@TYPE: Machine
#@NAME: quadmo747-x-iMx6 platform
#@DESCRIPTION: quadmo747-x-iMx6 platform configuration

require conf/machine/include/tune-cortexa9.inc

KERNEL_IMAGETYPE = "uImage"

PREFERRED_PROVIDER_virtual/kernel = "linux-quadmo-q7"
```

File to include. Contains processor params for the compiler

Type of kernel to build

Kernel package to use

If multiple recipes provide an item, this variable determines which recipe should be given preference.
Kernel recipe

Add kernel recipe:
meta-seco/recipes-kernel/linux/linux-quadmo-q7_3.0.35.bb

DESCRIPTION = "3.0 Linux Development Kernel for quadmo747-x-iMx6 board."
SECTION = "kernel"
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=..."
SRCPREV = "..."
SRC_URI = "git://com.example/sw/monitor/linux-3.0.35-QUADMO-iMX6;protocol=ssh \
file://imx6_seco_q7_defconfig.patch \
file://seco_cpld_n.patch"

S = "${WORKDIR}/git"

The name of the section in which packages should be categorized. Can be used by Package management.

Patches
Kernel recipe (cont)

Add kernel recipe:

meta-seco/recipes-kernel/linux/linux-quadmo-q7_3.0.35.bb

inherit kernel

do_configure()
{
    yes '' | oe_runmake imx6_seco_q7_defconfig
}

Override bitbake configuration step
Build the image and Deploy (uSD)

Build

`bitbake monitor-image`

Create this partition on uSD:

- EXT3 partition

```
tar -xvf build/tmp/deploy/images/quadmo747xiMx6/monitor-image.tar.gz
   -C path_of_the_usd_partition
```
References

https://www.yoctoproject.org/
https://www.yoctoproject.org/downloads/bsps
https://www.youtube.com/watch?v=zNLYanJAQ3s
https://www.yoctoproject.org/sites/default/files/elc-e_devday_introyocto_2.pdf