Intel CV Academy

OpenCV для встраиваемых платформ

Vitaly Tuzov

intel®

Зачем

- Интернет вещей
- Умный дом
- Носимые устройства

Современный мир ориентирован на маленькие устройства удобные для использования, но не для разработки.

Почему Yocto

Даже на функциональной платформе разработка может оказаться непростой задачей

RaspberryPI + Raspbian + native toolchain

Сложности:

- Распространение результата работы требует создания пакета
- Сборка занимает много времени
- Сборка требует существенного дискового пространства(1.8GB образ системы, 2.6GB после сборки)
- Подходит не для всех проектов

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install git cmake build-essentials
$ mkdir tests; cd tests
$ git clone https://github.com/opencv/opencv.git
$ mkdir opencv_build; cd opencv_build
$ cmake -DCMAKE_BUILD_TYPE=Release \
   -DBUILD_EXAMPLES=YES ../opencv/
$ make all
```

Yocto Project

- Yocto Project - проект посвященный созданию инструментария подготовки специализированных дистрибутивов для встраиваемых систем
  - http://www.yoctoproject.org/
- Проект основывается на OpenEmbedded – системе сборки(кросс-компиляции) пакетов для встраиваемых систем
  - http://www.openembedded.org/

Yocto Project

Достоинства

- Архитектурная независимость
- Переносимость кода
- Бинарная воспроизводимость
- Возможность создания персонализированных образов системы
- Гибкость разработки
- Развитый инструментарий
- Послойная структура разработки
- Поддержка частичного построения

Недостатки

- Высокий порог вхождения
- Нестандартная структура проекта требует привыкания
- Меньшая скорость разработки и тестирования
- Внесение изменений часто требует глубокого знакомства с документацией
- Большое время первого построения проекта

Система управления сборкой

Для управления сборкой используется утилита BitBake

- Image – готовый образ системы(прошивка), собранный из отдельных Package
- Package – некий полезный результат процедуры построения
- Package – результат обработки Recipe системой BitBake “baked recipe”
- Recipe – набор настроек и заданий, необходимых для построения одного или нескольких Package

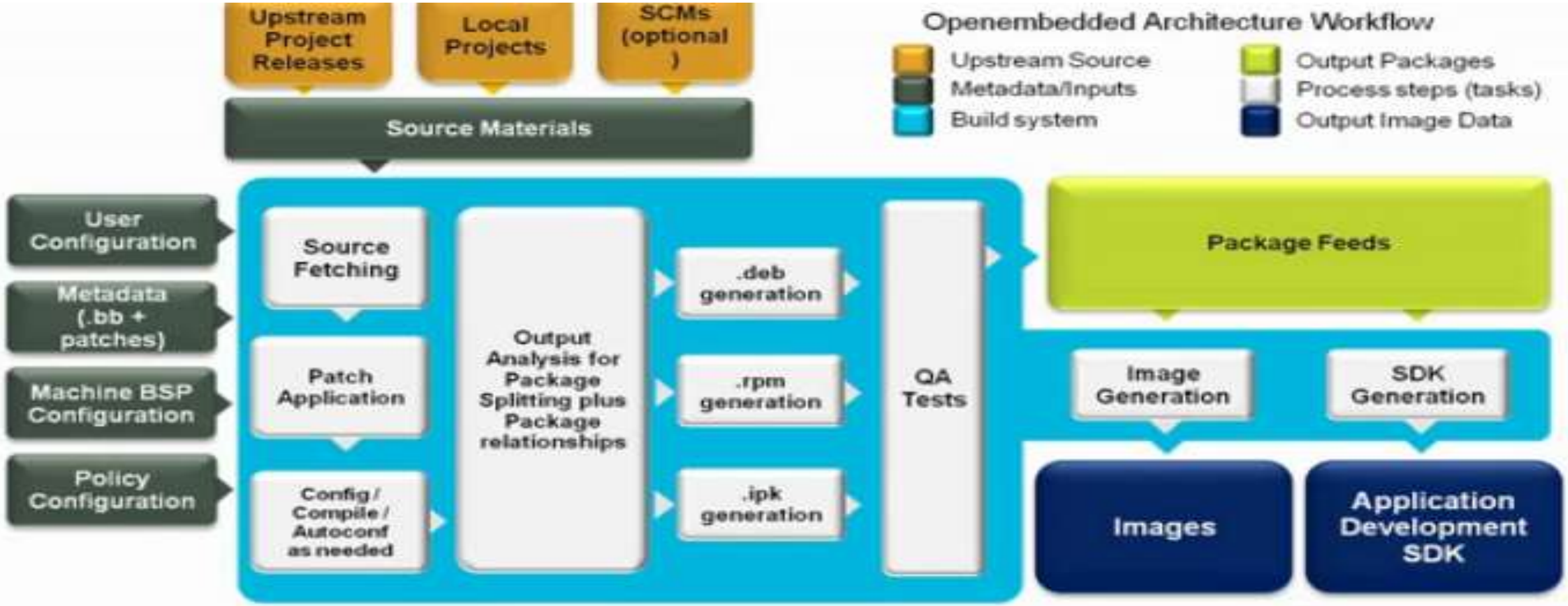
Слои

- Layers – наборы рецептов имеющих общее назначение и область применения
- Возможные области разделения слоев: BSP, интерфейс пользователя, настройки дистрибутива, утилиты, пользовательский приложения
- Слои могут содержать изменения и дополнения к уже существующим рецептам и настройкам

Структура рецепта

1. Fetch – получить исходный код
2. Extract – распаковать исходный код
3. Patch – внести локальные исправления
4. Configure – настроить окружение компиляции
5. Build – собрать бинарные файлы пакета
6. Install – сформировать итоговую структуру пакета
7. Package – подготовить итоговый пакет

Структура процесса построения



Немного практики

Лучше один раз увидеть

Готовим систему сборки

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install python git cmake chrpath texinfo
$ git clone git://git.yoctoproject.org/poky
$ cd poky
$ git clone git://git.yoctoproject.org/meta-raspberrypi
$ git clone git://git.openembedded.org/meta-openembedded.git
$ source oe-init-build-env rpi_build
```

Конфигурируем образ системы

```
$ vi conf/local.conf
```

MACHINE ?= "raspberrypi3“

DISTRO_FEATURES += "usbhost"

VIDEO_CAMERA = "1"

BB_NUMBER_THREADS = "4"

PARALLEL_MAKE = "-j4"

CORE_IMAGE_EXTRA_INSTALL += “opencv opencv-samples"

Настраиваем слои и строим образ

```
$ bitbake-layers add-layer ../meta-openembedded/meta-oe
$ bitbake-layers add-layer ../meta-raspberrypi
$ bitbake core-image-base
```

Результат готовая прошивка core-image-base-raspberrypi3.wic.bz2

Итоговый размер образа после сборки 270M

Под капотом

/meta-raspberrypi/recipes-bsp/u-boot/u-boot_%.bbappend

Коррекция к */poky/meta/recipes-bsp/u-boot.bb* добавляющая специфические для RaspberryPI настройки загрузчика

В */meta-raspberrypi/recipes-core/images/rpi-basic-image.bb*

IMAGE_INSTALL += " kernel-modules “

Коррекция для параметра в */poky/meta/classes/core-image.bbclass*

Под капотом

В *opencv_4.5.1.bb*

```
do_unpack_extra() {
    ...
}

addtask unpack_extra after do_unpack before do_patch
```


intel