Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет

по предмету «Системное программирование»

на тему:

«yocto»

Выполнил: студент группы № 63501/3

Дедков С.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е. В.

Санкт-Петербург

2015

Цель работы:

Изучить предоставляемые Yocto project шаблоны, инструменты и методы для создания специальных дистрибутивов Linux для встраиваемых систем на базе различных аппаратных архитектур.

Описать проект, инструменты и возможности.

Описать процесс сборки дистрибутива.

Собрать и запустить дистрибутив Linux для x86.

Собрать и запустить дистрибутив Linux для ARM.

Описание проекта, инструментов и возможностей.

Будучи проектом совместного сотрудничества (иногда такие проекты называются "зонтичными"), Yocto Project охватывает различные составляющие процесса разработки. Эти составляющие именуются проектамив рамках общего проекта Yocto Project и включают в себя инструменты для сборки, метаданные инструкций по сборке (называемые рецептами), библиотеки, утилиты и графические интерфейсы.

Инструменты:

* Poky – это эталонная система сборки в рамках проекта Yocto Project. Она включает в себя BitBake, OpenEmbedded-Core, пакет поддержки платформы (Board Support Package, BSP), а также прочие пакеты и компоненты, объединенные в единую сборку. Название Poky также относится к эталонному дистрибутиву Linux, который создается этой системой сборки и может быть чрезвычайно минималистичным (core-image-minimal) или же представлять собой полноценную систему Linux с графической оболочкой (core-image-sato).
* Набор метаданных разделен на слои, каждый из которых обладает дополнительной функциональностью по отношению к нижележащим слоям. Базовый слой называется OpenEmbedded-Core (или oe-core) и содержит общие рецепты, классы и связанные с ними функции, необходимые для любой сборки. Эти сборки впоследствии можно настраивать под собственные нужды, добавляя новые слои поверх слоя oe-core.
* Пакет поддержки платформы содержит пакеты и драйверы, необходимые для создания Linux-дистрибутива для определенной платформы или архитектуры. Эти пакеты часто поддерживаются производителями компьютерного оборудования. BSP-пакеты являются интерфейсом между операционной системой Linux и аппаратной частью, на которой она запускается. Заметим, что можно также создавать BSP для виртуальных машин.
* BitBake – это система сборки. Она считывает рецепты (определенные наборы инструкций) и следует им – скачивает необходимые пакеты, компилирует их и создает результирующие загрузочные образы. BitBake совместно поддерживается проектами Yocto Project и OpenEmbedded.
* Hob. Для упрощения процесса разработки Linux для встраиваемых устройств в проекте Yocto Project было реализовано несколько различных методов, позволяющих работать с графической средой. Относительно новым дополнением к проекту является Hob. Это дополнение предоставляет в распоряжение разработчиков графический пользовательский интерфейс для BitBake, т. е. для процесса сборки. Оба этих компонента постоянно развиваются с учетом отзывов пользователей.
* Embedded GLIBC (EGLIBC) – это вариант библиотеки GNU C Library (GLIBC), который был разработан для использования во встраиваемых системах. Особенностями EGLIBC являются меньший объем, настраиваемые компоненты и улучшенная поддержка кросс-компиляции и кросс-тестирования. EGLIBC входит в состав Yocto Project, но поддерживается своей собственной руководящей командой.
* Инструментарий для разработки приложений (Application Development Toolkit, ADT) позволяет разработчикам систем включать наборы SDK в создаваемые ими дистрибутивы с помощью инструментов Yocto Project. Впоследствии эти наборы SDK могут использовать сторонние разработчики для создания приложений для этих дистрибутивов. В состав ADT входят инструменты кросс-компиляции, утилиты для отладки и анализа производительности, а также сценарии эмуляции и поддержки QEMU. Кроме того в ADT включен подключаемый модуль Eclipse, который могут использовать те, кто предпочитает работать в интегрированной среде разработки (IDE).
* Autobuilder: предназначен для автоматизации тестов и оценки качества продуктов, развиваемых на базе Yocto Project.
* Cross-Prelink: предназначен для предварительной компоновки в средах, использующих кросс-компиляцию, что позволяет повысить производительность программ.
* Pseudo: эмулирует доступ от имени пользователя root, что необходимо при создании конечного загрузочного образа.
* Swabber: определяет, не содержит ли сборка, выполненная с использованием кросс-компиляции, компоненты хостовой системы.
* Build Appliance: это виртуальная машина, в которой запущен Hob; позволяет получить представление о Yocto Project из первых рук тем, кто создает сборки на компьютерах под управлением операционной системы, отличной от Linux (Примечание: на сегодняшний день инструменты сборки Yocto Project можно использовать только в ОС Linux).

Описание процесса сборки дистрибутива.

Загрузка инструментов:

Для загрузки poky можно воспользоваться двумя путями:

1. Можно загрузить tar файл с последней протестированной версией Yocto Project со страницы загрузки проекта

$ wget \

http://downloads.yoctoproject.org/releases/yocto/yocto-1.2/poky-denzil-7.0.tar.bz2

$ tar xjf poky-denzil-7.0.tar.bz2

$ cd poky-denzil-7.0

1. Можно получить последнюю версию (или любую отдельную ветку) при помощи git, хотя главная ветка разработки может оказаться менее стабильной по сравнению с протестированной версией, содержащейся в tar файле.

$ git clone git://git.yoctoproject.org/poky.git

$ cd poky

Инициализация рабочего окружения:

Сначала нужно инсталлировать все необходимые пакеты из репозитория программного обеспечения.

$ sudo apt-get install sed wget subversion git-core coreutils \

unzip texi2html texinfo libsdl1.2-dev docbook-utils fop gawk \

python-pysqlite2 diffstat make gcc build-essential xsltproc \

g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev \

autoconf automake groff libtool xterm libxml-parser-perl

Задание значений переменных окружения командной оболочки с помощью входящего в состав поставки сценария:

$ cd poky

$ . ./oe-init-build-env

Создание образа:

После задания значений переменных окружения рабочей директорией станет /build

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

You can now run 'bitbake <target>'

Common targets are:

core-image-minimal

core-image-sato

meta-toolchain

adt-installer

meta-ide-support

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

Выберем core-image-sato.

Далее можно задать настройки сборки в файле conf/local.conf

По умолчанию введены следующие настройки:

MACHINE ??= "qemux86"

DISTRO ?= "poky"

PACKAGE\_CLASSES ?= "package\_rpm"

EXTRA\_IMAGE\_FEATURES = "debug-tweaks"

USER\_CLASSES ?= "buildstats image-mklibs"

PATCHRESOLVE = "noop"

BB\_DISKMON\_DIRS = "\

STOPTASKS,${TMPDIR},1G,100K \

STOPTASKS,${DL\_DIR},1G,100K \

STOPTASKS,${SSTATE\_DIR},1G,100K \

STOPTASKS,/tmp,100M,100K \

ABORT,${TMPDIR},100M,1K \

ABORT,${DL\_DIR},100M,1K \

ABORT,${SSTATE\_DIR},100M,1K \

ABORT,/tmp,10M,1K"

PACKAGECONFIG\_append\_pn-qemu-native = " sdl"

PACKAGECONFIG\_append\_pn-nativesdk-qemu = " sdl"

ASSUME\_PROVIDED += "libsdl-native"

CONF\_VERSION = "1"

Для оптимизации сборки можно задать следующие параметры, связанные с параллельной обработкой, для ускорения процесса сборки. На этом шаге задаим значения этих параметров, равными двукратному числу ядер процессора (например, 8 для 4-ядерного процессора).

BB\_NUMBER\_THREADS = "8"

PARALLEL\_MAKE = "-j 8"

Введем команду для сборки. Это сборка с GUI SATO:

bitbake core-image-sato

После этого запустится сборка дистрибутива.

Далее для запуска образа нужно будет ввести команду:

runqemu qemux86