Building Images with Yocto Project- Lab

Tim Orling - Yocto Project, Intel Open Source Technology Center

March 9, 2018

These lab instructions are written for the *Building Images with Yocto Project* tutorial of the *Embedded Apprentice Linux Engineer* track. They are designed to work for the *PocketBeagle* hardware platform.

Initial build environment configuration

To get started with the Yocto Project, one of the easiest things to do is to clone the *poky* repository which will enable us to build the *poky* reference distribution.

The metadata layers at `https://github.com/e-ale/yocto-e-ale` provide a layers to work with *poky* to build a complete Linux distribution for the *PocketBeagle* platform. Let us go ahead and clone those now:

```
cd ..
git clone https://github.com/yocto-e-ale
```

However, for educational purposes, we are going to start straight from the official vanilla *poky*, and build our configuration from scratch.

Getting poky

Start by cloning poky from the official Yocto Project Git repository:

```
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky/
```

(Note: if download speed is too slow, you can use the `poky.tar.xz` tarball provided by the instructor)

We'll base our work on "master", which is the latest development branch. This branch will be released soon as the *2.5* or *sumo* release of the Yocto Project. Once it is released, an official *yocto-2.5* tag will become available and a *sumo* branch will be created which will provide on-going support for future dot releases (e.g. 2.5.1). As of this writing the project is in milestone three, or *M3* of the 2.5 release.

Setting up our local configuration

We can now setup a build environment and begin by building an image for *qemuarm*:

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

By default, the above command will build images for the *qemux86* machine, so we will need to alter the default configuration to enable the *qemuarm* machine, by uncommenting (removing the leading # on the following line in `conf/local.conf`:

```
#MACHINE ?= "qemuarm"
```

To save build and download time, we will use *shared state* and *downloads* already prepared for you by your instructor. Shared state, also known as *sstate* or *sstate-cache*, is a specially organized directory or cache of all the *tasks* that have been run to build a particular package or image. For teams, like ours, that are rebuilding the same set of packages, this is a way to dramatically speed up build time. The downloads directory can also be shared to save *fetch* time, or download time from the internet. Change the following lines in `conf/local.conf`:

```
#DL_DIR ?= "${TOPDIR}/downloads"
#SSTATE_DIR ?= "${TOPDIR}/sstate-cache"
```

to

```
DL_DIR ?= "${HOME}/DOWNLOADS"
SSTATE_DIR ?= "${HOME}/SSTATE"
```

We also might want to build *debian* packages, like the *BeagleBoard.org* stock images. To do so we need to change the following line in `conf/local.conf`:

```
PACKAGE_CLASSES ?= "package_rpm"
```

to

```
PACKAGE_CLASSES ?= "package_deb"
```

Also, to help speed up the build and save disk space, we can prevent temporary work being saved to disk by appending the following to `conf/local.conf`:

```
INHERIT += "rm_work"
```

Building a minimal image

To build a minimal image, with a very basic text console environment, we run the following command:

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

This will perform all the tasks necessary to fetch the source code, build the native host tools, compile and package the individual pieces of the image, and generate the root file system and the final image.

Running our image in emulation

We can test the result of our efforts by running the image in the *qemu* emulated environment, without graphics and using non-privileged network devices:

```
runqemu nographic slirp
```

Because the default setting is `EXTRA_IMAGE_FEATURES ?= "debug-tweaks"`, we can simply login as the *root* user with no password. This is only for development purposes and you should never ship a product with such an insecure setting.

Notice that the text before the login prompt represents the "branding" of the *poky* distribution:

```
Poky (Yocto Project Reference Distro) 2.4+snapshot qemuarm /dev/ttyAMA0

qemuarm login:
```

To change this default behavior, we will want to create our own custom Linux distribution, by creating a distro layer. We also want to add specific support for the *PocketBeagle*, so we will need to create a bsp layer. Finally, we will want to add some of our *recipes*, so we should create an application layer where we can put our work.

Create a distro layer

NOTE: if time is short or you want to skip ahead, the content of this section can be cloned by running:

```
$ cd ../yocto-e-ale
$ git checkout 01_create_distro_layer
```

Bitbake comes with a tool to create the directory structure of a generic metadata layer. We will use this tool now to create our distro layer skeleton.

```
bitbake-layers create-layer ../yocto-e-ale/meta-e-ale-distro
```

Let's look at what was created:

```
meta-e-ale-distro
├─ conf
│  └─ layer.conf
├─ COPYING.MIT
├─ README
└─ recipes-example
   └─ example
      └─ example.bb
```

This is a generic metadata layer, more like an application or functional layer, and does not have the pieces that make it a distro layer. We will now delete what we do not need and add what is missing.

```
$ cd meta-e-ale
$ rm -rf recipes-example
$ mkdir -p conf/distro/include
$ vim conf/distro/e-ale.conf
```

Add the following content:

```
DISTRO = "e-ale"
DISTRO_NAME = "e-ale Linux"

DISTRO_VERSION = "1.0+snapshot-${DATE}"

E_ALE_DEFAULT_DISTRO_FEATURES = "systemd"

LAYER_CONF_VERSION ?= "7"

# Add ssh server so we can connect to system running this image remotely,
# add development tools (gcc, make, etc), and -dev packages for installed #
E_ALE_DEFAULT_DISTRO_FEATURES += "ssh-server-openssh tools-sdk dev-pkgs"

DISTRO_FEATURES ?= "${DISTRO_FEATURES_DEFAULT} ${DISTRO_FEATURES_LIBC} ${E_

VIRTUAL-RUNTIME_init_manager = "systemd"

INHERIT += "uninative"

UNINATIVE_URL = "http://downloads.yoctoproject.org/releases/uninative/1.7/"
UNINATIVE_CHECKSUM[i686] ?= "de51bc9162b07694d3462352ab25f636a6b50235438c1b
UNINATIVE_CHECKSUM[x86_64] ?= "ed033c868b87852b07957a4400f3b744c00aef5d6470
```

In order to add our branding before the login prompt, we need to enable our local changes to override the default in the `base-files` recipe:

```
$ mkdir -p recipes-core/base-files
$ pushd recipes-core/base-files
$ echo '# look for files in this layer first' > base-files_%.bbappend
$ echo 'FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"' >> base-files_%.bbapp
```

This tells *Bitbake* to look at our `meta-e-ale-distro` layer before any other layer of lower priority (allows us to override default behavior). The := operator says to immediately expand the variables `${THISDIR}` and `${PN}`. The : at the end is important and easy to miss, with that your paths will be mangled in a non-functional way.

Next, we want to add our branding to the file which will be installed in our root file system at `/etc/issue`:

```
$ mkdir recipes-core/base-files/base-files
$ vim recipes-core/base-files/base-files/issue
```

We want our result to look like the following:

```
                  _        _     _
                 | |      | |   (_)
  ___        __ _| | ___  | |    _ _ __  _   ___  __
 / _ \______/ _` | |/ _ \ | |   | | ’_ \| | | \ \/ /
|  __/_____| (_| | |  __/ | |___| | | | | |_| |>  <
 \___|      \__,_|_|\___| \_____/_|_| |_|\__,_/_/\_\

Embedded Apprentice Linux Engineer http://e-ale.org
```

Unfortunately, we need to escape all the backslash characters, so the actual content is less readable in the file itself:

```
                  _        _     _
                 | |      | |   (_)
  ___        __ _| | ___  | |    _ _ __  _   ___  __
 / _ \\______/ _` | |/ _ \\ | |   | | ’_ \\| | | \\ \\/ /
|  __/_____| (_| | |  __/ | |___| | | | | |_| |>  <
 \\___|      \\__,_|_|\\___| \\_____/_|_| |_|\\__,_/_/\\_\\

Embedded Apprentice Linux Engineer http://e-ale.org
```

We also want branding for network access (like ssh):

```
$ pushd recipes-core/base-files/base-files
$ cp issue issue.net
$ popd
```

Configuration Templates

We would like to make it easy to start working with our distro layer, so we will provide templates for `local.conf` and `bblayers.conf`. These are created in the `meta-e-ale-disro/conf` directory as `local.conf.sample` and `bblayers.conf.sample`.

For `local.conf.sample`, we can simply copy our `local.conf` from our prior session and just change the distro from "poky" to "e-ale":

```
$ cp ../build-qemuarm/conf/local.conf \
   meta-e-ale-distro/conf/local.conf.sample
$ sed -i -e ’s:DISTRO = "poky":DISTRO = "e-ale":g’ \
   meta-e-ale-distro/conf/local.conf.sample
```

For `bblayers.conf.sample`, the template has content that is automatically replaced, so the syntax is a bit trickier and we use the following content:

```
# LAYER_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
LCONF_VERSION = "1"

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

BBLAYERS ?= " \
  ##OEROOT##/meta \
  ##OEROOT##/../yocto-e-ale/meta-e-ale-distro \
  "
```

The special tokens `##OEROOT##` are substituted (with the full path to openembedded-core) automatically when the template is used to create the `bblayers.conf` file. For now, we only include the distro layer we are currently working on. We will add more layers in the next sections. Be careful with the paths that you use, as you need to remember that this file will be sourced in the context of the environment variable `${OEROOT}`, which in our usage will be set to be the "poky" directory by our `init-build-env` script. We will populate that script now.

Rather than using the default *openembedded-core* init script contained in *poky*, we will use our own that will setup the build environment for our distro (and in the next section, board support for the *PocketBeagle*).

Create the build environment initialization script `init-build-env` in the `yocto-e-ale` directory with the following content:

```
!/bin/sh

# OE Build Environment Setup Script
#
# Copyright (C) 2006-2011 Linux Foundation
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

#
# Normally this is called as '. ./oe-init-build-env <builddir>'
#
# This works in most shells (not dash), but not all of them \
pass the arguments
# when being sourced.  To workaround the shell limitation \
use "set <builddir>"
# prior to sourcing this script.
#
# LAYER_CONF_VERSION is increased each time \
build/conf/bblayers.conf
# changes incompatibly
LCONF_VERSION = "7"
    THIS_SCRIPT_DIR=$(dirname "$THIS_SCRIPT")
    THIS_SCRIPT_DIR=$(readlink -f "$THIS_SCRIPT_DIR")

    if [ -d "$THIS_SCRIPT_DIR/meta-e-ale-distro/conf" ]; then
        TEMPLATECONF="$THIS_SCRIPT_DIR/meta-e-ale-distro/conf"
    elif [ -f "$THIS_SCRIPT_DIR/.templateconf" ]; then
        source $THIS_SCRIPT_DIR/.templateconf
    elif [ -d "$THIS_SCRIPT_DIR/.template" ]; then
        TEMPLATECONF="$THIS_SCRIPT_DIR/.template"
    fi
fi
```

continued:

```
if [ -z "$OEROOT" ]; then
    OEROOT=$(dirname "$THIS_SCRIPT")/../poky
    OEROOT=$(readlink -f "$OEROOT")
fi

export OEROOT

if [ -z "$BITBAKEDIR" ]; then
    BITBAKEDIR=$(dirname "$THIS_SCRIPT")/../poky/bitbake
    BITBAKEDIR=$(readlink -f "$BITBAKEDIR")
fi

export BITBAKEDIR

unset THIS_SCRIPT_DIR
unset THIS_SCRIPT

. $OEROOT/scripts/oe-buildenv-internal &&
    TEMPLATECONF="$TEMPLATECONF" \
    $OEROOT/scripts/oe-setup-builddir || {
    unset OEROOT
    return 1
}
unset OEROOT

[ -z "$BUILDDIR" ] || cd "$BUILDDIR"

# Shutdown any bitbake server if the BBSERVER variable is not set
if [ -z "$BBSERVER" ] && [ -f bitbake.lock ]; then
    grep ":" bitbake.lock > /dev/null && BBSERVER=$(cat \
    bitbake.lock) bitbake --status-only
    if [ $? = 0 ]; then
        echo "Shutting down bitbake memory resident server \
        with bitbake -m"
        BBSERVER=$(cat bitbake.lock) bitbake -m
    fi
fi
```

We can now create a new build environment with our new distro layer:

```
$ . ./init-build-env ../build-e-ale-1
$ bitbake core-image-minimal
$ runqemu nographic slirp
```

The text at the login will now be what we want for our branding:

```
                   _        _      _
                  | |      | |    (_)
  ___         __ _| | ___  | |     _ _ __  _   ___  __
 / _ \______/ _` | |/ _ \ | |    | | ’_ \| | | \ \/ /
|  __/______| (_| | |  __/ | |____| | | | | |_| |>  <
 \___|       \__,_|_|\___| \______/_|_| |_|\__,_/_/\_\

Embedded Apprentice Linux Engineer http://e-ale.org
e-ale Linux 1.0+snapshot qemuarm ttyAMA0

qemuarm login:
```

Just to quickly review, our directory structure should now look like the following:

```
yocto-e-ale
├─ init-build-env
└─ meta-e-ale-distro
   ├─ conf
   │  ├─ bblayers.conf.sample
   │  ├─ distro
   │  │  └─ e-ale.conf
   │  ├─ layer.conf
   │  └─ local.conf.sample
   ├─ COPYING.MIT
   ├─ README
   └─ recipes-core
      └─ base-files
         ├─ base-files
         │  ├─ issue
         │  └─ issue.net
         └─ base-files_%.bbappend
```

(More to come...)