Final Project – Linux build for Raspberry Pi based on Yocto

For EECS_X497.10/LEC/1(FALL 2013,UNEX,00103)

By Karl Stiller
Introduction

For this final project I attempted a build for the raspberry pi board using the Yocto build system. I planned the project with three stages which were:

- 1. Build and test of old but known working versions of Yocto and Raspberry Pi layer.
- 2. Build and test of the latest version of Yocto and Raspberry Pi layer.
# Raspberry Pi Board (Model B)

<table>
<thead>
<tr>
<th>Chip</th>
<th>Broadcom BCM2835 SoC full HD processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>700 MHz Low Power ARM1176JZ-F Applications Processor</td>
</tr>
<tr>
<td>GPU</td>
<td>Dual Core VideoCore IV® Multimedia Co-Processor</td>
</tr>
<tr>
<td>Memory</td>
<td>512MB SDRAM</td>
</tr>
<tr>
<td>Ethernet</td>
<td>onboard 10/100 Ethernet RJ45 jack</td>
</tr>
<tr>
<td>Video</td>
<td>HDMI, Composite RCA</td>
</tr>
<tr>
<td>Audio</td>
<td>3.5mm jack, HDMI</td>
</tr>
<tr>
<td>Storage</td>
<td>SD, MMC, SDIO card slot</td>
</tr>
<tr>
<td>Size</td>
<td>8.6cm x 5.4cm x 1.7cm</td>
</tr>
</tbody>
</table>
Required hardware

• HDMI to DVI conversion cable
• 5v 700mA (min) Phone charger
• SD card
• Raspberry Pi
• Anti-Idiot protector box (optional)
• Monitor and USB keyboard (not shown)
Introduction to Yocto

• Yocto is a prefix meaning 10^-24. Something very tiny indeed.

• “The Yocto Project is an open-source collaboration project focused on embedded Linux developers. The Yocto Project uses a build system based on the Poky project to construct complete Linux images. The Poky project, in turn, draws from and contributes back to the OpenEmbedded project.” [2]
Yocto build system

- Poky, the OpenEmbedded projects upon which Yocto is based, and Yocto itself is very mature. Yocto started in 2010.
- If you are building under Linux it's a simple matter of downloading the project, installing some necessary tools and you are ready to build an image.
- It is also possible to build an image using Windows and the Yocto Project Build Appliance.
- An overview of the system is shown on the next slide.
Yocto Project Development Environment

Overview of the Yocto build system [2]

- Click to edit Master text styles
  - Second level
  - Third level
  - Fourth level
  - Fifth level

The Yocto Project Development Environment
Part 1.

• To make a start on building an image for the Raspberry Pi, I followed the recipe that I found on a blog. (see [3])

• I am using a Debian based Linux distribution (Crunchbang) so I first installed the necessary tools for Debian.

    ```
sudo apt-get install gawk wget git-core \
diffstat unzip texinfo build-essential \
chrpath libsdl1.2-dev xterm
```
Part 1.

• Next step is to download Yocto, Poky and OpenEmbedded

$ git clone git://git.yoctoproject.org/poky yoctoProject
Cloning into 'yoctoProject'...
remote: Counting objects: 209629, done.
remote: Compressing objects: 100% (53241/53241), done.
remote: Total 209629 (delta 151862), reused 209317 (delta 151556)
Receiving objects: 100% (209629/209629), 97.17 MiB | 51 KiB/s, done.
Resolving deltas: 100% (151862/151862), done.

• Then checkout the known working version

$ cd yoctoProject
$ git checkout 4a36a32567ecfbc7ce7b967803e6e23314953ef5
Part 1

• Then download the Raspberry Pi Layer (do this in the yoctoProject folder created earlier)

$ git clone https://github.com/djwillis/meta-raspberrypi.git
Cloning into 'meta-raspberrypi'...
remote: Counting objects: 1059, done.
remote: Compressing objects: 100% (679/679), done.
remote: Total 1059 (delta 380), reused 893 (delta 284)
Receiving objects: 100% (1059/1059), 229.71 KiB | 39 KiB/s, done.
Resolving deltas: 100% (380/380), done.

• Checkout the known working version

$ cd meta-raspberrypi
$ git checkout 305c5259e24ea9fb285ca983dc4f9454743fa0c
Part 1.

• Returning to the yoctoProject folder, I setup the build variables using

   $ source oe-init-build-env raspberryPiBuild/

• This creates a new folder with default scripts which will need to be edited in order to create a raspberry pi image.
Part 1

• I changed or included the following lines in the local configuration file conf/local.conf (I am using a netbook as my host so the parallel build is limited):

  BB_NUMBER_THREADS = "2"
  PARALLEL_MAKE = "-j 2"
  MACHINE ?= "raspberrypi"
  BBMASK = "meta-raspberrypi/recipes-multimedia/libav|meta-raspberrypi/recipes-core/systemd"
  PACKAGE_CLASSES ?= "package_deb"

• I added the raspberrypi layer to the conf/bblayers.conf file

  BBLAYERS ?= "
    /mnt/yoctoProject/meta 
    /mnt/yoctoProject/meta-yocto 
    /mnt/yoctoProject/meta-yocto-bsp 
    /mnt/yoctoProject/meta-raspberrypi 
  "
Part 1

• I then ran the build command
  
  `bitbake rpi-basic-image`

• Then went to bed ☺️ On a netbook it took many hours to compile and build the image.

• After a good night sleep... I copied the image to a SD card using the following command:
  
  `$ dd if=tmp/deploy/images/raspberrypi/rpi-basic-image-raspberrypi.rpi-sdimg of=/dev/sdb`

  229376+0 records in
  229376+0 records out

  117440512 bytes (117 MB) copied, 67.9435 s, 1.7 MB/s
Part 1

• Result: Working System.

• Some detail:

Poky 8.0 (Yocto Project 1.3 Reference Distro) 1.3+snapshot-20131124 tty1
root@raspberrypi:~# uname -a
Linux raspberrypi 3.2.27 #1 PREEMPT Sun Nov 24 07:36:31 BRST 2013 armv6l GNU/Linux
root@raspberrypi:~# cat /proc/version
Linux version 3.2.27 (karl@crunchbang) (gcc version 4.7.2 (GCC) ) #1 PREEMPT Sun Nov 24 07:36:31 BRST 2013
root@raspberrypi:~# cat /etc/issue
Poky 8.0 (Yocto Project 1.3 Reference Distro) 1.3+snapshot20131124 \l
Problems encountered during Part 1

• Linux was allocated a measly 10G worth of space which I thought would have been adequate for the class, though I didn't think of the final project. Oops.

• I tried fixing this by mounting a thumb drive to a handly location (such as the tmp folder in the build directory) to extend the available space. However this proved to be a real pain to re-setup for each build and I ended up creating a partition with the recommended size of 32G just for the builds on my hard drive.
Problems encountered during Part 1

• I was receiving the following warning during the build, though I ignored it. It didn’t seem to cause any problems. This error disappeared in part 2.

WARNING: Host distribution "Debian GNU/Linux 7.2 (wheezy)" has not been validated with this version of the build system; you may possibly experience unexpected failures. It is recommended that you use a tested distribution.
Part 2.

• Build the bleeding edge version...

• I ran the following command to get back to the bleeding edge:

  karl@crunchbang:/mnt/yoctoProject$ git checkout master

  Previous HEAD position was 4a36a32... libpng: fix packaging

  Switched to branch 'master'

  karl@crunchbang:/mnt/yoctoProject$ cd meta-raspberrypi/

  karl@crunchbang:/mnt/yoctoProject/meta-raspberrypi$ git checkout master

  Previous HEAD position was 305c525... arch-arm: define different ARMPKGARCH when different CCARGS are used

  Switched to branch 'master'
Part 2 – Build #2

• I promptly ran into a whole pile of problems ❌

ERROR: Function failed: Fetcher failure: Fetch command failed with exit code 128, output:
fatal: reference is not a tree: ae937f99fee8a37f2ddd7270f6bcc0e497e8c903

• My first work around was simply to mask out the recipes that weren’t working by masking them out by adding the following in conf/local.conf

BBMASK = "meta-raspberrypi/recipes-multimedia/libav|meta-raspberrypi/recipes-core/systemd"
BBMASK .= "|meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi_3.10.18.bb"
BBMASK .= "|meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi_3.12.0.bb"
Part 2 – Build #2

• Rerunning the build command generated a working image

• Some detail:
  Poky (Yocto Project Reference Distro) 1.5+snapshot-20131126 raspberrypi /dev/tty1
  root@raspberrypi:~# uname -a
  Linux raspberrypi 3.11.7 #1 Tue Nov 26 00:18:03 BRST 2013 armv6l GNU/Linux
  root@raspberrypi:~# cat /proc/version
  Linux version 3.11.7 (karl@crunchbang) (gcc version 4.8.2 (GCC) ) #1 Tue Nov 26 00:18:03 BRST 2013
  root@raspberrypi:~# cat /etc/issuePoky (Yocto Project Reference Distro)
  1.5+snapshot20131126  
  

Part 2 – Build #3

- As I am an engineer and programmer (and a very stubborn one at that!) I felt that simply masking out a broken recipe was an easy way out. I also had quite some time left and decided that I would try and get it working.

- I cloned the git repository for the linux branch which was giving the problem and tried to find the revision it was referring to. In the history there were no references with the revision it was referring to.
• I searched for the error on Google and after reading a bit I guessed that it was probably due to the developer using a reference from his local git repository.

• On the [https://github.com/djwillis/meta-raspberrypi](https://github.com/djwillis/meta-raspberrypi) website it indicated to use the `bitbake rpi-hwup-image` command, tried it but gave me the same error (all future builds used this command). It also said that the repository had moved to [http://git.yoctoproject.org/git/meta-raspberrypi](http://git.yoctoproject.org/git/meta-raspberrypi). Tried this repository but the error was the same.

• Digging a bit deeper, I had a look at the recipe and saw that it was simply applying a patch to a single file - an i2c driver file. In the history of the git repository there was only one version of this patch, and only one version of the file being patched. I assumed that the head revision of that branch was probably going to work.

• I edited the recipe to fetch the master (head) reference of the linux repository
Part 2 – Build #3

**Success! A *new* different error! **

ERROR: Command Error: exit status: 1  Output:
Applying patch sl030raspberrypii2ckernel.patch
can't find file to patch at input line 14
Perhaps you used the wrong -p or --strip option?
The text leading up to this was:
... *(omitting patch for reasons of space)*
----No file to patch.  Skipping patch.
2 out of 2 hunks ignored
Patch sl030raspberrypii2ckernel.patch does not apply (enforce with -f)
ERROR: Function failed: patch_do_patch
ERROR: Logfile of failure stored in:
/mnt/yoctoProject/raspberryPiBuild/tmp/work/raspberrypi-poky-linux-gnueabi/linux-raspberrypi/3.12.0+gitmaster-r0/temp/log.do_patch.9795
NOTE: recipe linux-raspberrypi-3.12.0+gitmaster-r0: task do_patch: Failed
ERROR: Task 82 (/mnt/yoctoProject/meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi_3.12.0.bb, do_patch) failed with exit code '1'
• Somehow, the script couldn’t find the file to apply the patch. I looked at the patch in the repository I cloned, I looked at the driver file. It looked perfect. I compared the lines. They were. The line numbers were a little different. I changed them to be correct.

• Trying again it still would not patch the file. I looked at the driver file in the branch that was able to apply the patch and the branch that was not able to apply the patch. They were the same file, and all branches use the same patch file. The path looked correct. I pleaded with the build script to try again. It game me the same error.

• Sidestepping insanity I edited the recipe so that it wouldn’t patch the file.

• Building again – Success!
Part 2 – Build #3

• Some detail:

Poky (Yocto Project Reference Distro) 1.5+snapshot-20131202 raspberrypi /dev/tty1

root@raspberrypi:~# uname –a

Linux raspberrypi 3.11.7 #1 Tue Dec 2 20:20:13 BRST 2013 armv6l GNU/Linux

root@raspberrypi:~# cat /proc/version

Linux version 3.11.7 (karl@crunchbang) (gcc version 4.8.2 (GCC)) #1 Mon Dec 2 20:20:18 BRST 2013

root@raspberrypi:~# cat /etc/issue
Poky (Yocto Project Reference Distro) 1.5+snapshot20131202 


Build #4

• In build #3 it is using the 3.11.7 kernel even though it built the 3.12.0 version kernel. Tried forcing the latest kernel version, masking all recipes, leaving only the (3.12.0) version and applying the patch. I thought that maybe there was a conflict in patching all of these kernel versions. Patch still did not apply.

• I then looked at the git repository that yocto had cloned in the raspberryPiBuild/tmp/work/raspberrypi-poky-linux-gnueabihf/linux-raspberrypi/ directory. Searching for the i2c driver file I discovered that the local git repository had a problem. It was missing the whole drivers directory. Looking at the remote repository via a browser it is clearly there. I tried forcing the build to re-fetch the repository by deleting the linux-raspberrypi directory and relevant files and directories in raspberryPiBuild/downloads/git2/, but it would not re-fetch. As a last resort I deleted all files and directories in raspberryPiBuild, leaving only the conf folder. Ready for a fresh build.
Build #4

• Errors continued in the fresh build using the 3.12.0 version kernel. Though, a different error. Looking again at the local repository fetched by yocto, it again was missing some files. I don’t know why yocto wasn’t able to fetch the repository successfully and didn’t give any errors about this. Perhaps my partition is corrupted, my internet connection is bad, or perhaps the git server was overloaded at the time.

• I was now running out of time, and wouldn’t be able to do another fresh build. I decided to use the 3.11.7 kernel, hoping that the local repository for this version was complete. It was.
Part 3 – Build #4

• I continued on using build #4, as a base for part 3 which was to modify the build in some way. I decided on adding the perl package.

• To find out how to add a package to the generated image I had a look at the Yocto wiki[5]. Which said the easiest way was to add the following to the local.conf file in the local build folder:

```bash
IMAGE_INSTALL_append += " perl"
```
Build #4

• Building again and running the overused phrase at the prompt gave:

![Image showing terminal output]

• When using builds 1, 2 and 3 gave the following error, so perl was added successfully:

Perl: not found
Closing Remarks

• The Yocto build system is very easy to use and well documented, however for somebody just starting out the documentation is a little daunting.

• There is much to learn about the build system, I have only just started learning about this great system.

• If I were to redo this project I would use a faster computer, as the netbook struggled to compile the image, even with Linux installed natively. A fresh build on it took over 8 hours.

• It was a shame that the local repository of the linux source had a problem after spending so long to discover what the problem was. Perhaps reformatting the hard drive and using a network cable rather than Wi-Fi would be a good idea if I were to retry a fresh build in the future.
References

- [2] http://www.yoctopproject.org/docs/current/yocto-project-qs/yocto-
- [5] https://wiki.yoctopproject.org/wiki/FAQ#How_can_I_add_a_package