WRnode Cross-compilation on Raspberry Pi and Acadia

- @ StudentID: 3130000026
- @ Author skar.Weidtsps_skar@zju.edu.cn
- @ Date: 2015/04/17

ATTENTION!

The process is almost the same with the cross-compile setup on your PC. Yet the compressed toolchain file cannot be directly uncompressed to run on your board as the CPU(ISA) varies.

So we need to compile the toolchain on board.

```
$ cd YOUR-PREFERABLE-DIRECORY
$ wget http://d.wrtnode.com/sdk/sdk.tar.bz2
$ tar -jxvf sdk.tar.bz2
$ cd wrtnode-sdk
```

If you found any dependency required, use `sudo apt-get install to install them` .

Type `make menuconfig` in your terminal, and select toolchain to compile.

Note that the configuration automatically include the commands to compile the kernel and those packages, excluding them may help you to accelerate the compilation.

After you type `make menuconfig` the process starts collecting information of packages and information of target.

It may take a while before you move forward.

We may process ahead, without waiting. Type `make V=s`

Go eat a nice wholesome sandwich, drink a pop, call a friend, play a video game, and generally find something to do. That is to say, this process is going to take tedious period to finish. According to author's experience, it would probably take a night (from 1 a.m. to ~8 a.m.) on **Acadia** (with `-j4`), one single day on **Raspberry Pi** (without `-j` option)

Note that to accelerate the compilation a bit on **Raspberry Pi**, you may take a look at the **overlock** technology.

Results

use the complied toolchain under directory `wrtnode-sdk/staging_dir/` to test if we successfully compiled the toolchain.

First of all, `export` a new variable `wrtnode-sdk/staging_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin"` to $PATH

Unluckily I got this

```
$ mipsel-openwrt-linux-gcc hello.c -o hello
$ mipsel-openwrt-linux-gcc: warning: environment variable 'STAGING_DIR' not
defined
```

Seems that we would better `export STAGING_DIR="/home/linaro/wrtcross/wrtnode-sdk/staging_dir"` in our `.bashrc`

Then we successfully compile the c code "hello.c"
Type the following command and check the output result.

```
$ readelf -h hello
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 01 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       1
  Type:                              EXEC (Executable file)
  Machine:                           MIPS R3000
  Version:                           0x1
  Entry point address:               0x4004f0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          2904 (bytes into file)
  Flags:                             0x70001005, noreorder, cpic, o32,
mips32r2
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         8
  Size of section headers:           40 (bytes)
  Number of section headers:         34
  Section header string table index: 31
```

Reference

TechBlog