Day 02-Booting & Toolchains

What’s in the Beagle?

- Software
- What happens when the Beagle boots Linux?

```
Instruments X-Loader 1.4.4ss (Aug 19 2010 - 02:49:27)
Beagle xM Rev A
Reading boot sector
Loading u-boot.bin from mmc
```

What happens when the Beagle powers up?

```
U-Boot 2010.03-dirty (Aug 20 2010 - 20:50:46)

OMAP3630/3730-GP ES1.0, CPU-OPP2, L3-165MHz,
OMAP3 Beagle board + LPDDR/NAND
I2C:   ready
DRAM:  512 MB
NAND:  0 MiB
*** Warning - bad CRC or NAND, using default
environment

In:    serial
Out:   serial
Err:   serial
```

What happens when the Beagle powers up?

```
No EEPROM on expansion board
Beagle xM Rev C
Die ID #34780000061000000156166b0a02300a
Hit any key to stop autoboot:  0
mmc1 is available
The user button is currently NOT pressed.
reading boot.scr

687 bytes read
Running bootscript from mmc ...
## Executing script at 80200000
mmc1 is available
reading uImage

3193476 bytes read
```

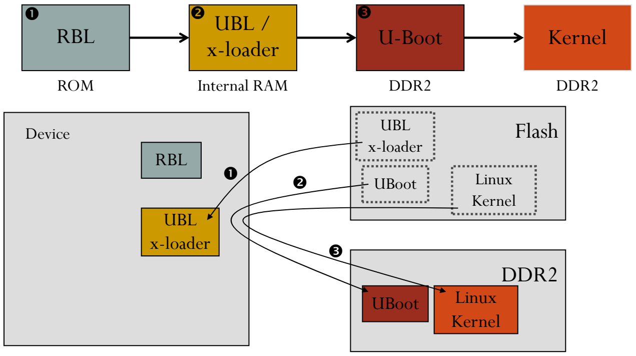
What happens when the Beagle powers up?

```
## Booting kernel from Legacy Image at 80200000 ...
   Image Name:   Angstrom/2.6.32/beagleboard
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3193412 Bytes =  3 MB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux...........................................................
[    0.000000] Linux version 2.6.32 (daniel@kids-laptop) (gcc version 4.3.3 (GC0
[    0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7f
[    0.000000] CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction ce
```

Booting Linux – ROM to Kernel



U-boot

- OMAP3 beagleboard.org # **print mmcboot**
  - mmcboot=echo Booting from mmc ...;
  - run mmcargs;
  - bootm ${loadaddr}
- OMAP3 beagleboard.org # **print mmcargs**
  - mmcargs=setenv bootargs console=${console} ${optargs} mpurate=${mpurate} buddy=${buddy} camera=${camera} vram=${vram} omapfb.mode=dvi:${dvimode} omapdss.def_disp=${defaultdisplay} root=${mmcroot} rootfstype=${mmcrootfs}

U-boot

- OMAP3 beagleboard.org # **run mmcargs**
- OMAP3 beagleboard.org # **print bootargs**
  - bootargs=console=ttyS2,115200n8 mpurate=1000 buddy=none camera=lbcm3m1 vram=12M omapfb.mode=dvi:640x480MR-16@60 omapdss.def_disp=dvi root=/dev/mmcblk0p2 rw rootfstype=ext3 rootwait

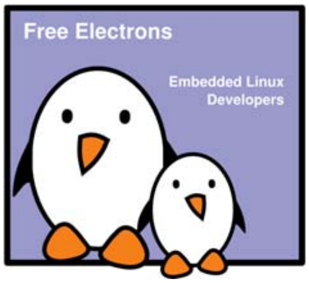
Development Environment

- Cross-Compiling

Embedded Linux system development

Cross-compiling toolchains

Thomas Petazzoni
Michael Opdenacker
Free Electrons

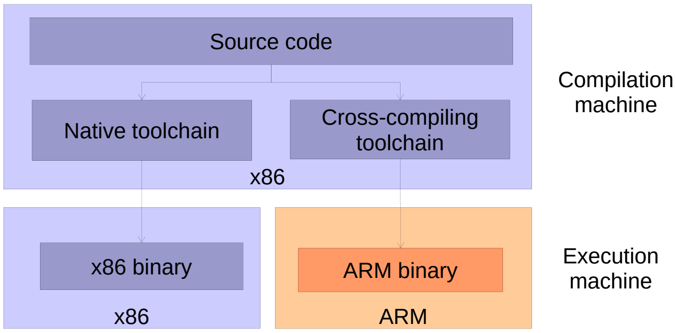


Latest update: 1/13/2012,
Document sources, updates and translations:
http://free-electrons.com/docs/toolchains
Corrections, suggestions, contributions and translations are welcome!

Definition (1)

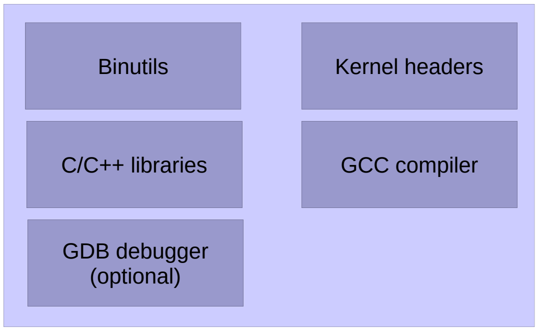
- The usual development tools available on a GNU/Linux workstation is a **native toolchain**
- This toolchain runs on your workstation and generates code for your workstation, usually x86
- For embedded system development, it is usually impossible or not interesting to use a native toolchain
  - The target is too restricted in terms of storage and/or memory
  - The target is very slow compared to your workstation
  - You may not want to install all development tools on your target.
- Therefore, **cross-compiling toolchains** are generally used. They run on your workstation but generate code for your target.

Definition (2)



DSP too!

Components



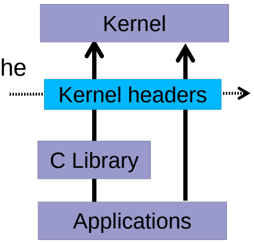
binutils

- **Binutils** is a set of tools to generate and manipulate binaries for a given CPU architecture
- **as**, the assembler, that generates binary code from assembler source code
- **ld**, the linker
- **ar**, **ranlib**, to generate .a archives, used for libraries
- **objdump**, **readelf**, **size**, **nm**, **strings**, to inspect binaries. Very useful analysis tools !
- **strip**, to strip useless parts of binaries in order to reduce their size
- http://www.gnu.org/software/binutils/
- GPL license

Kernel headers (1)

- The C library and compiled programs needs to interact with the kernel
  - Available system calls and their numbers
  - Constant definitions
  - Data structures, etc.
- Therefore, compiling the C library requires kernel headers, and many applications also require them.

MAY1



- Available in <linux/...> and <asm/...> and a few other directories corresponding to the ones visible in include/ in the kernel sources

Slide 25

MAY1 I don't know where these are.

Maybe /usr/include
Mark A. Yoder, 12/22/2009

GCC compiler

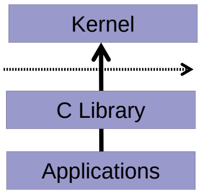
- GNU C Compiler, the famous free software compiler
- Can compile C, C++, Ada, Fortran, Java, Objective-C, Objective-C++, and
- Generate code for a large number of CPU architectures, including **ARM**, AVR, Blackfin, CRIS, FRV, M32, MIPS, MN10300, PowerPC, SH, v850, i386, x86_64, IA64, Xtensa, etc.
- http://gcc.gnu.org/
- Available under the GPL license, libraries under the LGPL.

C library

- The C library is an essential component of a Linux system
- Interface between the applications and the kernel
- Provides the well-known standard C API to ease application development

MAY3

- Several C libraries are available: glibc, uClibc, eglibc, dietlibc, newlib, etc.
- The choice of the C library must be made at the time of the cross-compiling toolchain generation, as the GCC compiler is compiled against a specific C library.



Slide 29

MAY3 application programming interface
Mark A. Yoder, 12/22/2009

glibc

http://www.gnu.org/software/libc/

- License: LGPL
- C library from the GNU project
- Designed for performance, standards compliance and portability
- Found on all GNU / Linux host systems
- Of course, actively maintained
- Quite big for small embedded systems: approx 2.5 MB on arm (version 2.9 - libc: 1.5 MB, libm: 750 KB)

uClibc

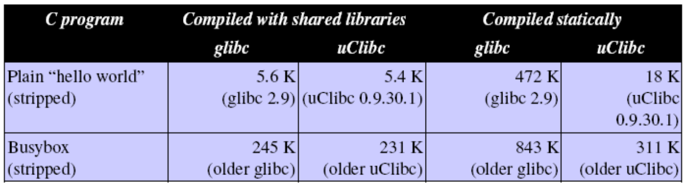
http://www.uclibc.org/ from CodePoet Consulting

- License: LGPL
- Lightweight C library for small embedded systems
- High configurability: many features can be enabled or disabled through a menuconfig interface
- Works only with Linux/uClinux, works on most embedded architectures
- No stable ABI, different ABI depending on the library configuration
- Focus on size rather than performance
- Small compile time

uClibc (2)

- Most of the applications compile with uClibc. This applies to all applications used in embedded systems.
- Size (arm): 4 times smaller than glibc!
  uClibc 0.9.30.1: approx. 600 KB (libuClibc: 460 KB, libm: 96KB)
  glibc 2.9: approx 2.5 MB
- Used on a large number of production embedded products, including consumer electronic devices
- Actively maintained, large developer and user base
- Now supported by MontaVista, TimeSys and Wind River.

Honey, I shrunk the programs!



C program	*Compiled with shared libraries*		*Compiled statically*	
	glibc	*uClibc*	*glibc*	*uClibc*
Plain “hello world” (stripped)	5.6 K (glibc 2.9)	5.4 K (uClibc 0.9.30.1)	472 K (glibc 2.9)	18 K (uClibc 0.9.30.1)
Busybox (stripped)	245 K (older glibc)	231 K (older uClibc)	843 K (older glibc)	311 K (older uClibc)

Executable size comparison on ARM

eglibc

« Embedded glibc », under the LGPL

- Variant of the GNU C Library (GLIBC) designed to work well on embedded systems
- Strives to be source and binary compatible with GLIBC
- eglibc's goals include reduced footprint, configurable components, better support for cross-compilation and cross-testing.
- Can be built without support for NIS, locales, IPv6, and many other features.
- Supported by a consortium, with Freescale, MIPS, MontaVista and Wind River as members.
- The Debian distribution is switching to eglibc too: http://blog.aurel32.net/?p=47
- http://www.eglibc.org

Other smaller C libraries

- Several other smaller C libraries have been developed, but none of them have the goal of allowing the compilation of large existing applications
- They need specially written programs and applications
- Choices :
- Dietlibc, http://www.fefe.de/dietlibc/. Approximately 70 KB.
- Newlib, http://sources.redhat.com/newlib/
- Klibc, http://www.kernel.org/pub/linux/libs/klibc/, designed for use in an initramfs or initrd at boot time.

Building a toolchain (3)

- Many decisions must be made when building a toolchain
  - Choosing the C library
  - Choosing the version of the different components
  - Choosing the configuration of the toolchain
  - Which ABI should be used ? Toolchains for the ARM architecture for example, can generate binaries using the OABI (Old ABI) or the EABI (Embedded ABI), that are incompatible
  - Should the toolchain support software floating point, or does the hardware support floating point operations ?
  - Should the toolchain support locales, IPv6, or other specific features ?

Get a precompiled toolchain

- Solution that most people choose, because it is the simplest and most convenient solution
- First, determine what toolchain you need: CPU, endianism, C library, component versions, ABI, soft float or hard float, etc.
- Many toolchains are freely available pre-compiled on the Web
- CodeSourcery, http://www.codesourcery.com, is a reference in that area, but they only provide glibc toolchains.
- See also http://elinux.org/Toolchains

Installing and using a precompiled toolchain

- Follow the installation procedure proposed by the vendor
- Usually, it is simply a matter of extracting a tarball at the proper place
- Then, add the path to toolchain binaries in your PATH: export PATH=/path/to/toolchain/bin/:$PATH

Toolchain building utilities (2)

- Crosstool
  - The precursor, written by Dan Kegel
  - Set of scripts and patches, glibc-only
  - Not really maintained anymore
  - http://www.kegel.com/crosstool
- Crosstool-ng
  - Rewrite of Crosstool, with a menuconfig-like configuration system
  - Feature-full: supports uClibc, glibc, eglibc, hard and soft float, many architectures
  - Actively maintained
  - http://ymorin.is-a-geek.org/dokuwiki/projects/crosstool

Toolchain building utilities (3)

Many root filesystem building systems also allow the construction of cross-compiling toolchain

- Buildroot
  - Makefile-based, uClibc only, maintained by the community
  - http://buildroot.uclibc.org
- PTXdist
  - Makefile-based, uClibc or glibc, maintained mainly by Pengutronix
  - http://www.pengutronix.de/software/ptxdist/index_en.html
- OpenEmbedded
  - The feature-full, but complex building system
  - http://www.openembedded.org/