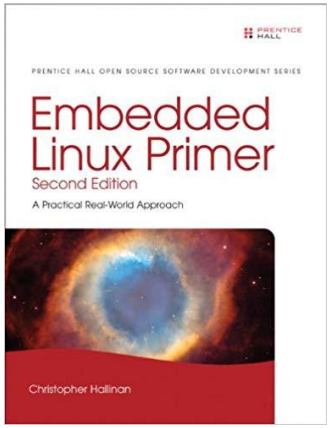
Based on RPi & Embedded Linux Primer 2nd Edition

Embedded Linux



Date: 2018.07.02
Author: Slowboot
Email: chunghan.yil@gmail.com
Doc Revision:0.9

이번 Course에서 배울 내용

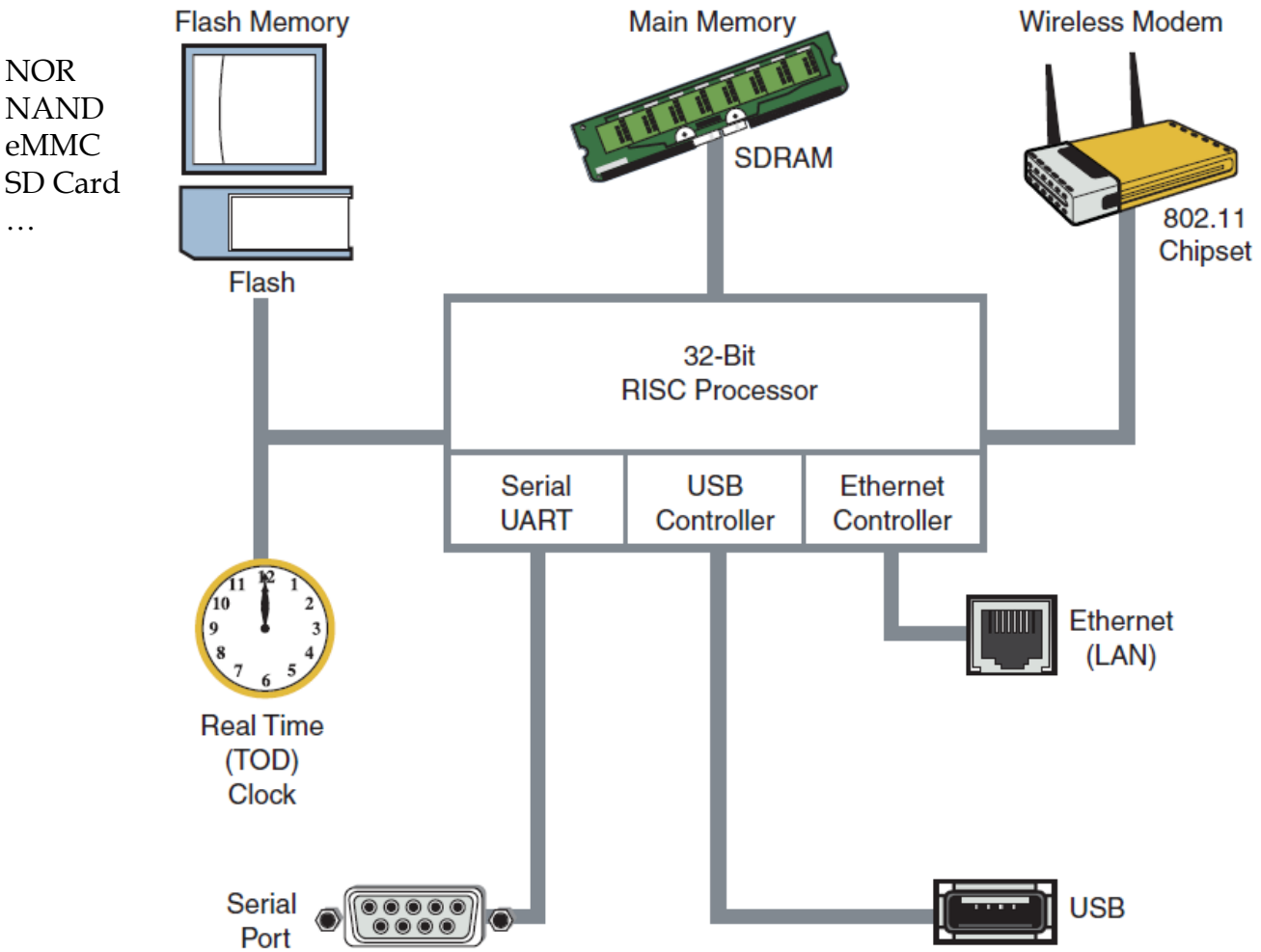
- 1. Raspberry Pi3에 대한 이해
- 2. ARM Toolchain 사용법
- 3. Bootloader & u-boot build 방법
- 4. Device Tree 개요 이해
- 5. Linux Kernel & build 방법
- 6. Linux kernel module programming
- 7. Buildroot & build 방법
- …

목차

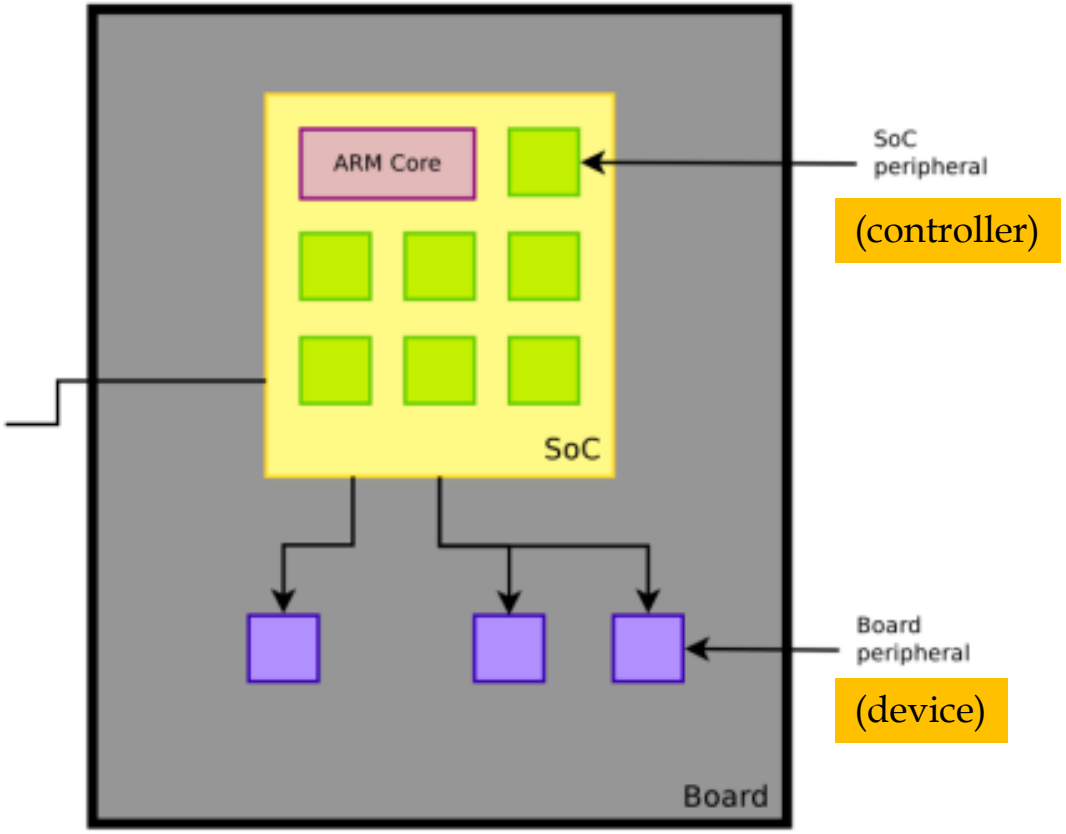
Chapter 1 – 3

: Embedded System 개요

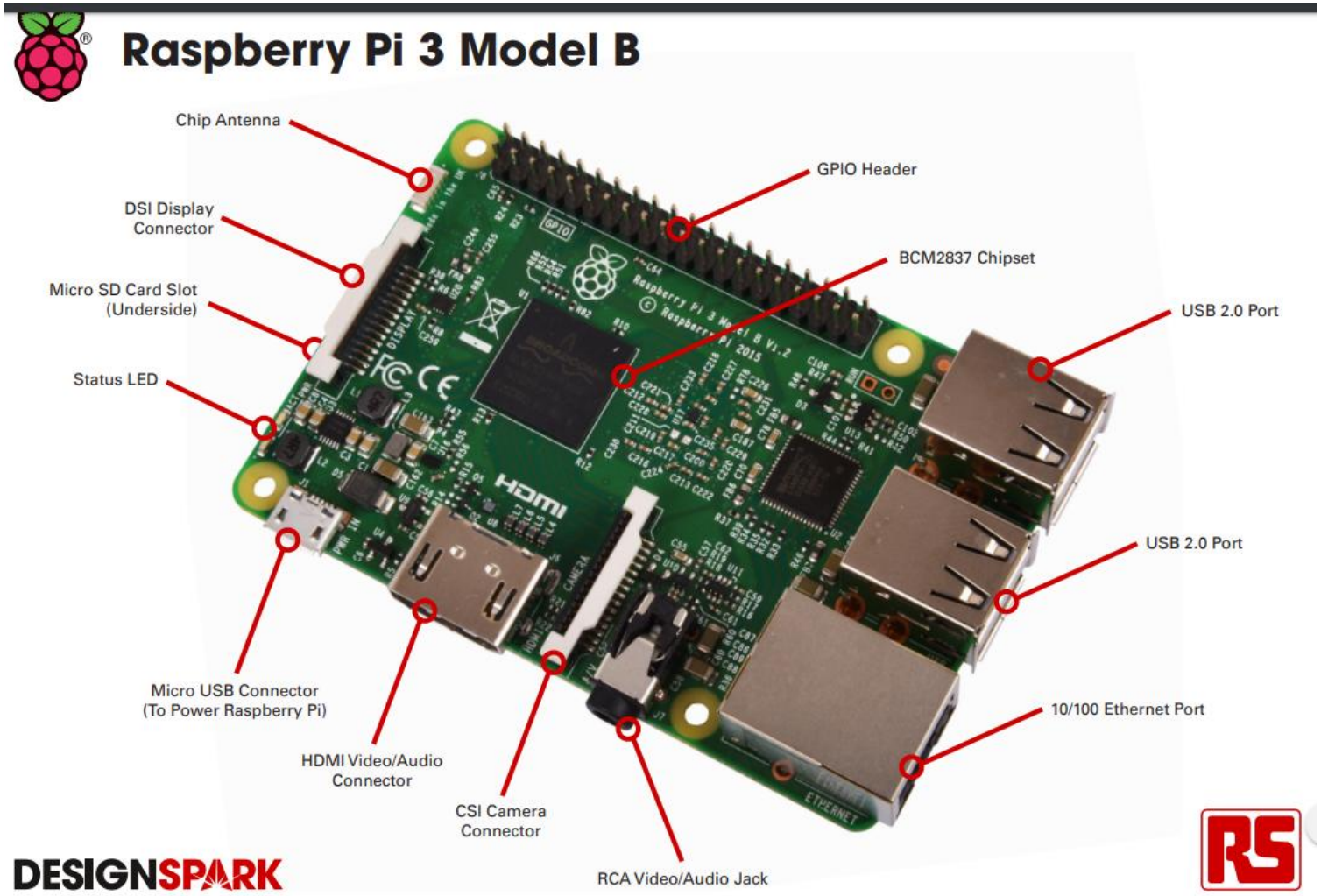
1. 일반적인 Embedded System Block도(1)



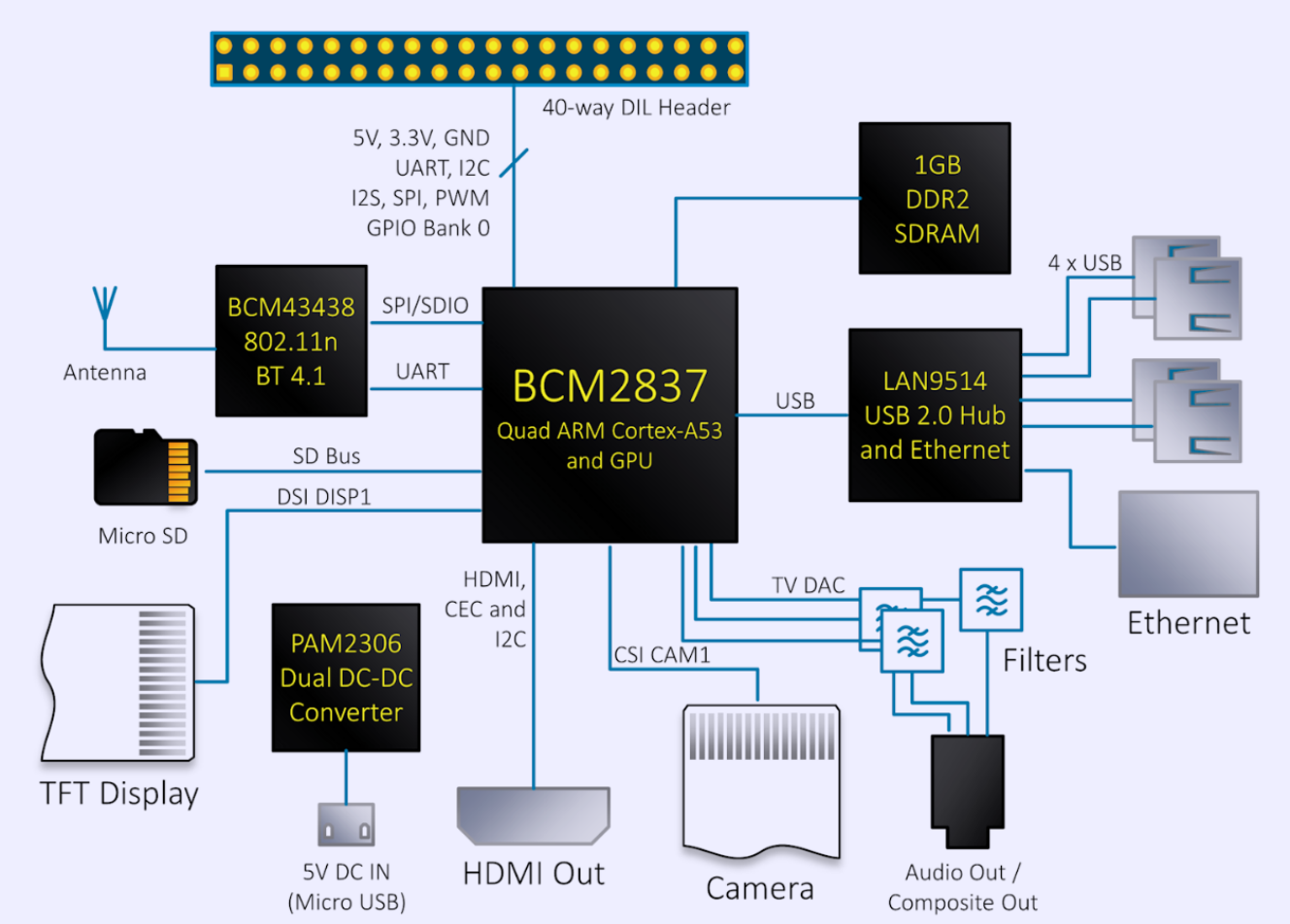
1. 일반적인 Embedded System Block도(2) – ARM SoC & Board



2. Raspberry Pi 3(1) – 보드 & GPIO 헤더(1)



2. Raspberry Pi 3(1) – 보드 & GPIO 헤더(2)



2. Raspberry Pi 3(1) – 보드 & GPIO 헤더(3)

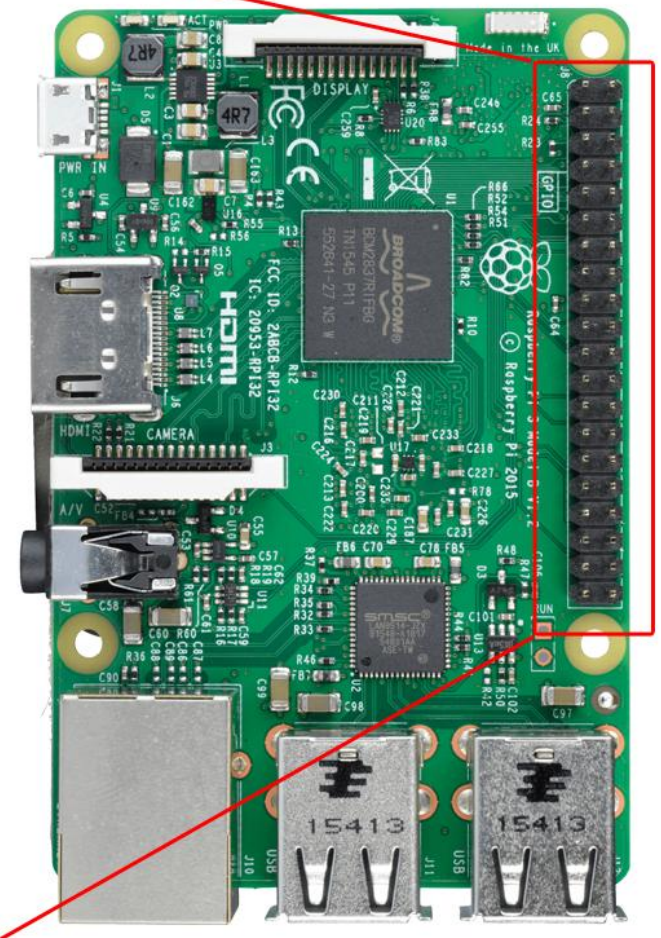
GPIO: General Purpose I/O

Raspberry Pi 3 Model B (J8 Header)

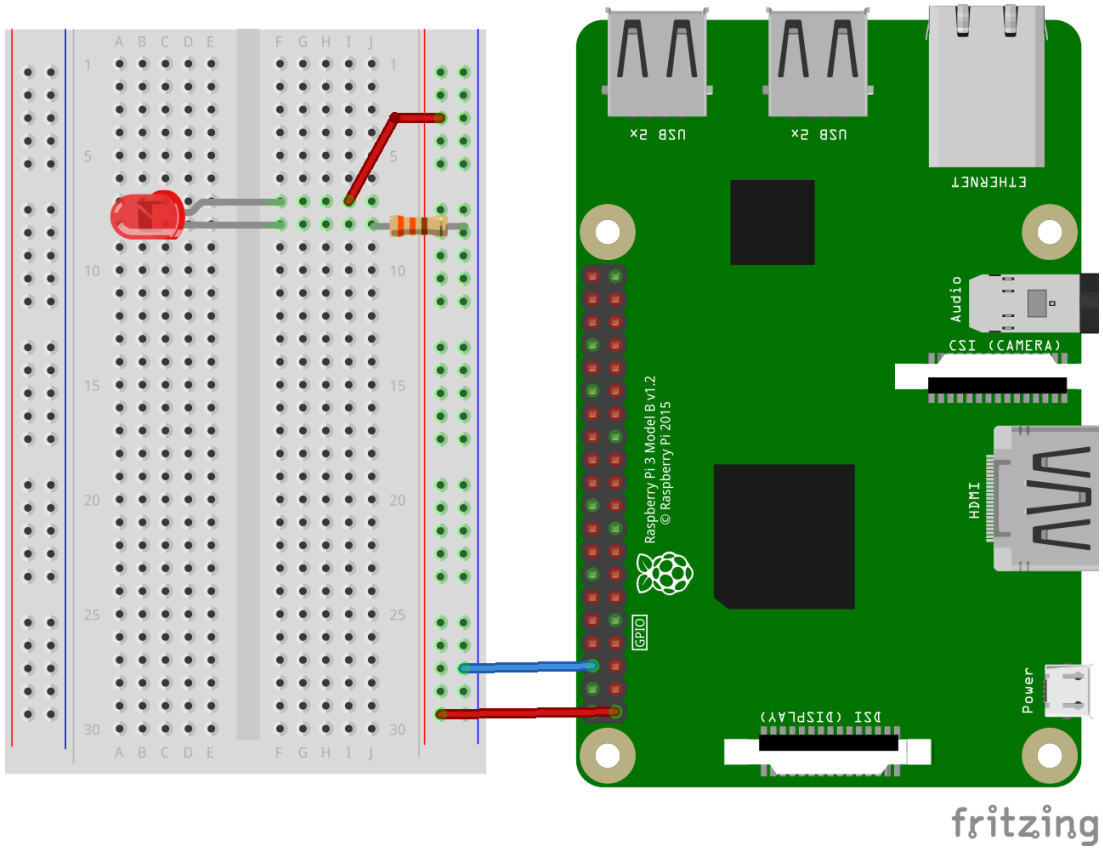
GPIO#	NAME				NAME	GPIO#
	3.3 VDC Power	1	2		5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	4		5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5	6		Ground	
7	GPIO 7 GPCLK0	7	8		GPIO 15 TxD (UART)	15
	Ground	9	10		GPIO 16 RxD (UART)	16
0	GPIO 0	11	12		GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	14		Ground	
3	GPIO 3	15	16		GPIO 4	4
	3.3 VDC Power	17	18		GPIO 5	5
12	GPIO 12 MOSI (SPI)	19	20		Ground	
13	GPIO 13 MISO (SPI)	21	22		GPIO 6	6
14	GPIO 14 SCLK (SPI)	23	24		GPIO 10 CE0 (SPI)	10
	Ground	25	26		GPIO 11 CE1 (SPI)	11
30	SDA0 (I2C ID EEPROM)	27	28		SCL0 (I2C ID EEPROM)	31
21	GPIO 21 GPCLK1	29	30		Ground	
22	GPIO 22 GPCLK2	31	32		GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	34		Ground	
24	GPIO 24 PCM_FS/PWM1	35	36		GPIO 27	27
25	GPIO 25	37	38		GPIO 28 PCM_DIN	28
	Ground	39	40		GPIO 29 PCM_DOUT	29

Attention! The GIPO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

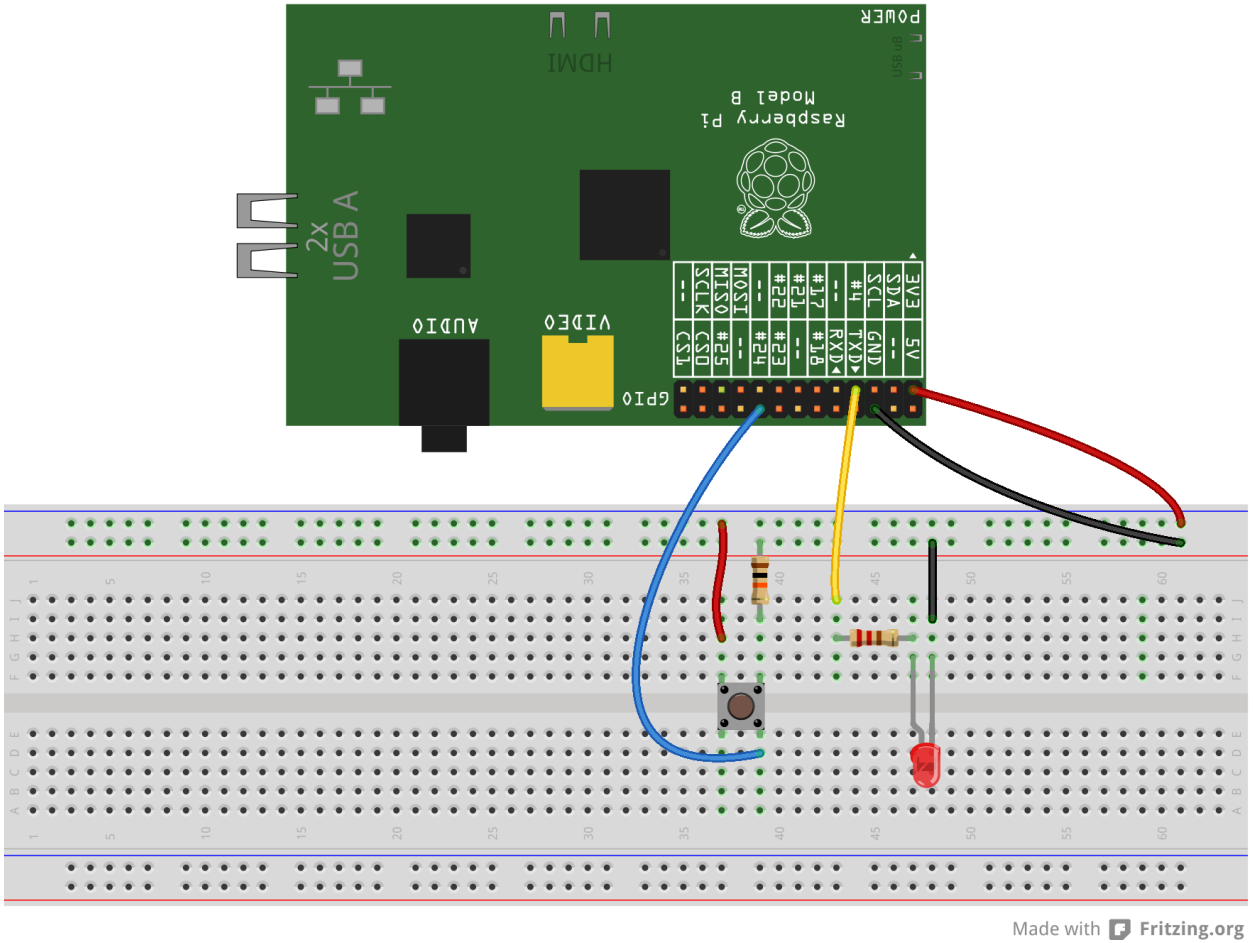
http://www.pi4j.com



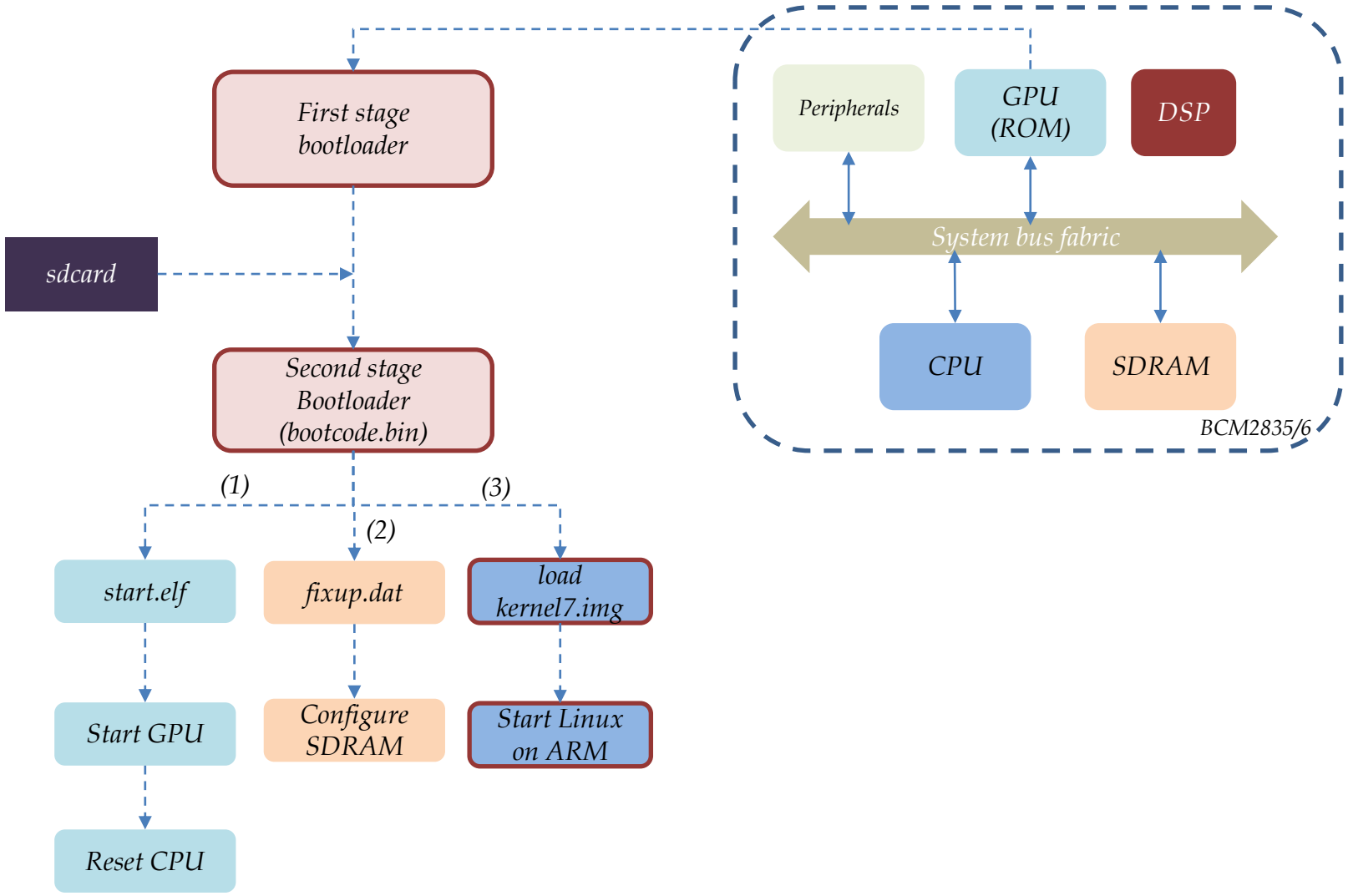
2. Raspberry Pi 3(1) – 보드 & GPIO 헤더(4)



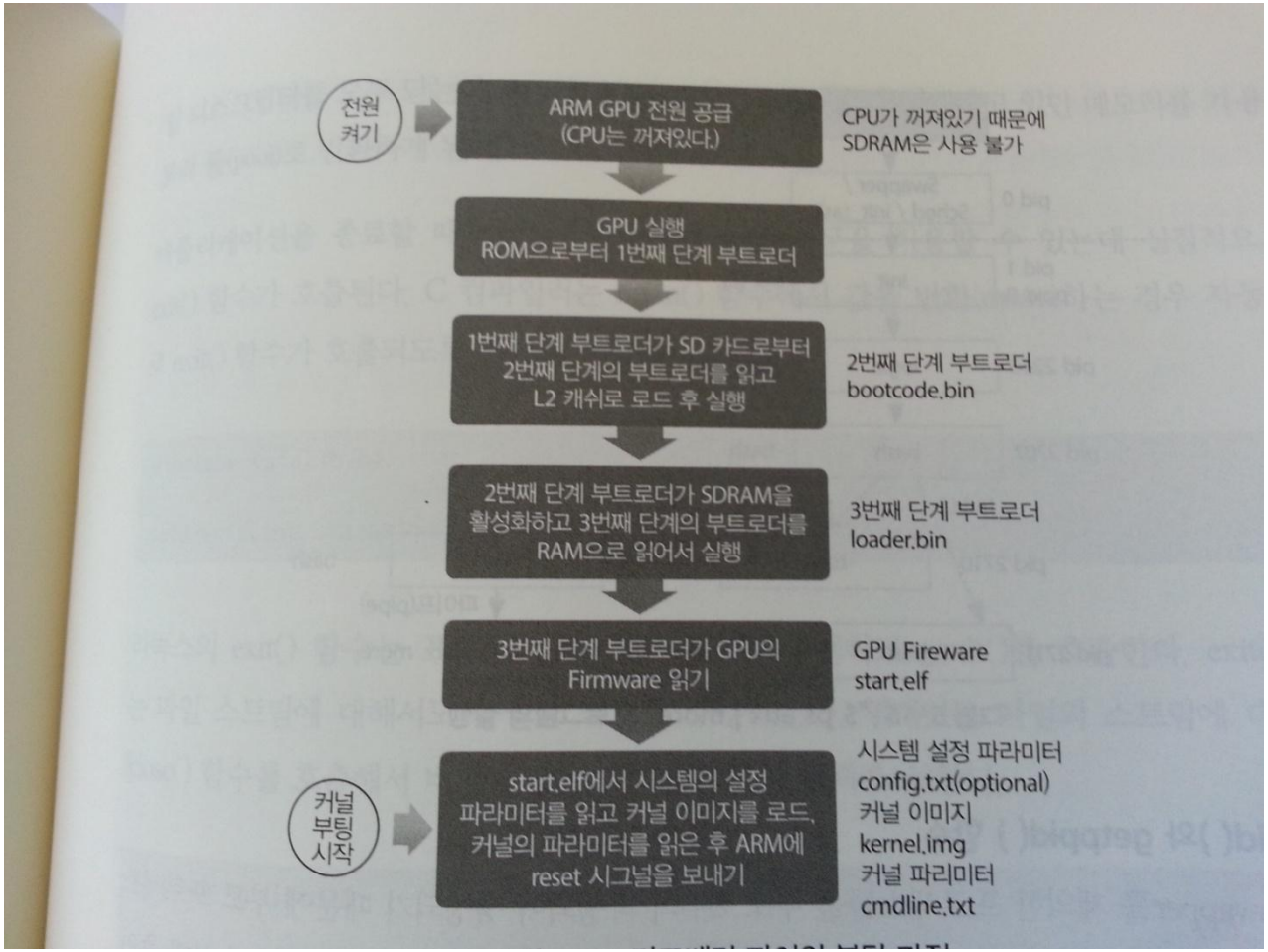
2. Raspberry Pi 3(1) – 보드 & GPIO 헤더(5)



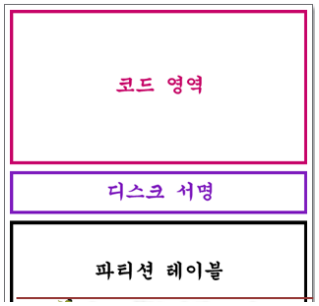
2. Raspberry Pi 3(2) - Booting 순서(1)



2. Raspberry Pi 3(2) - Booting 순서(2)



2. Raspberry Pi 3(3) - microSD card 구성



MBR(512 byte)

첫 번째 파티션

Bootable fat32
(bootloader, dtb, kernel)

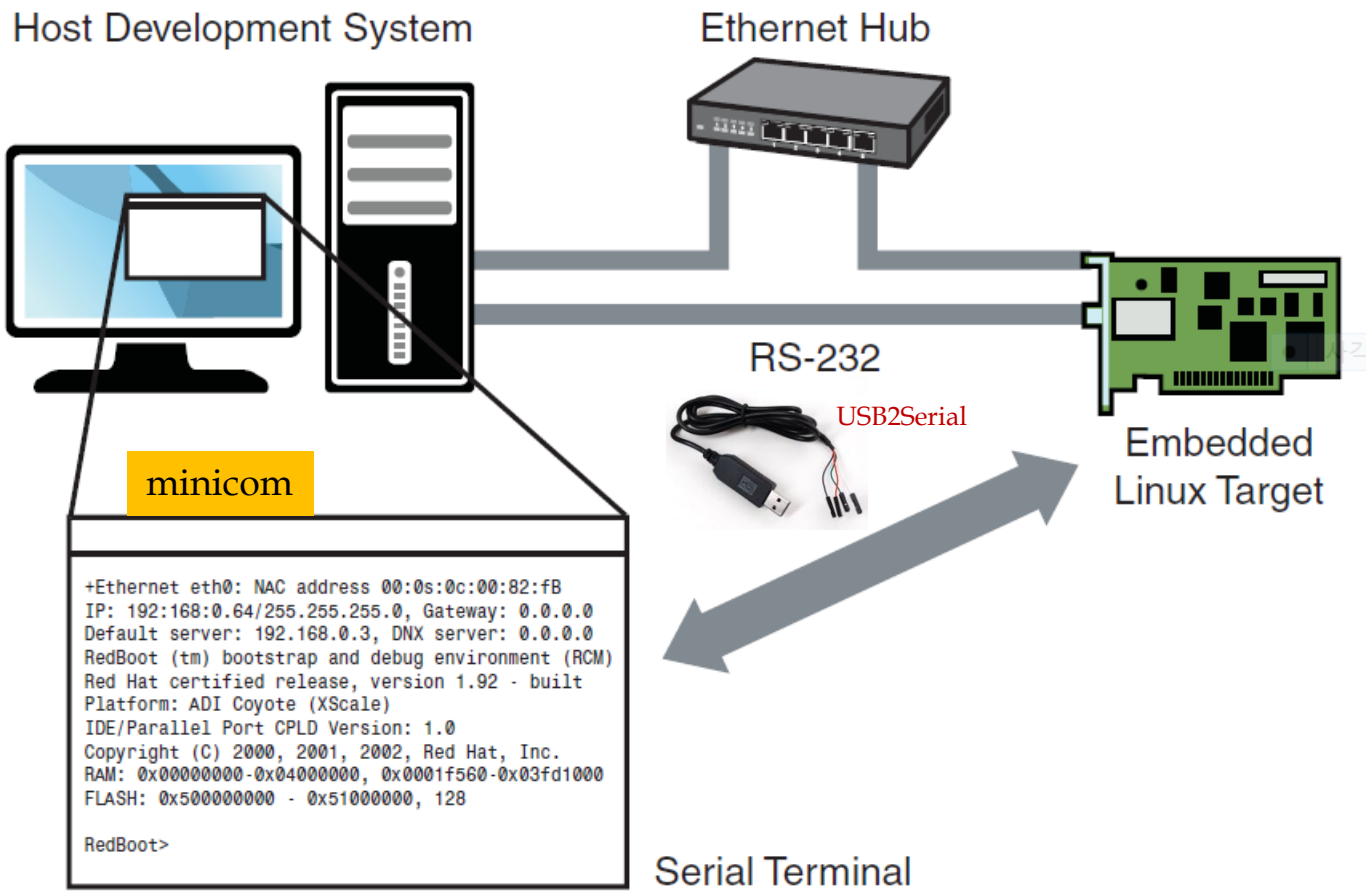
두 번째 파티션

Rootfs 영역

bootcode.bin
config.txt
cmdline.txt
start.elf
fixup.dat
bcm2710-rpi-3-b.dtb
overlays/
kernel7.img(= zImage)

/
+-bin/
+-dev/
+-etc/
+- home/
+-lib32-> lib/
+-linuxrc -> bin/busybox
+-media/
+-mnt/
+-opt/
+-proc/
+-root/
+-run/
+-sbin/
+-sys/
+-tmp/
+-usr/
+-var/

3. Embedded System(1) - Host Computer & Target Board



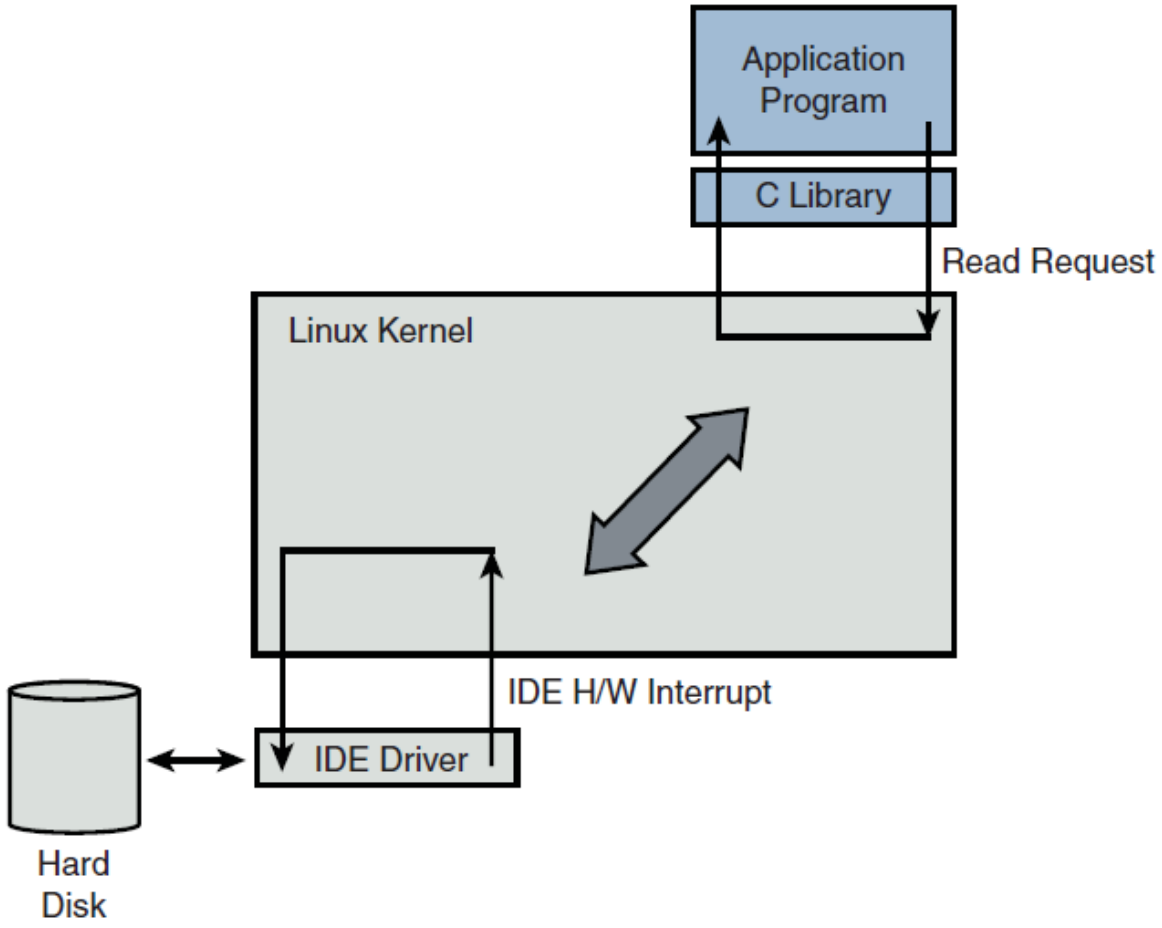
3. Embedded System(2) – minicom

- $ minicom –s
- $ picocom –b 115200 /dev/ttyUSB0

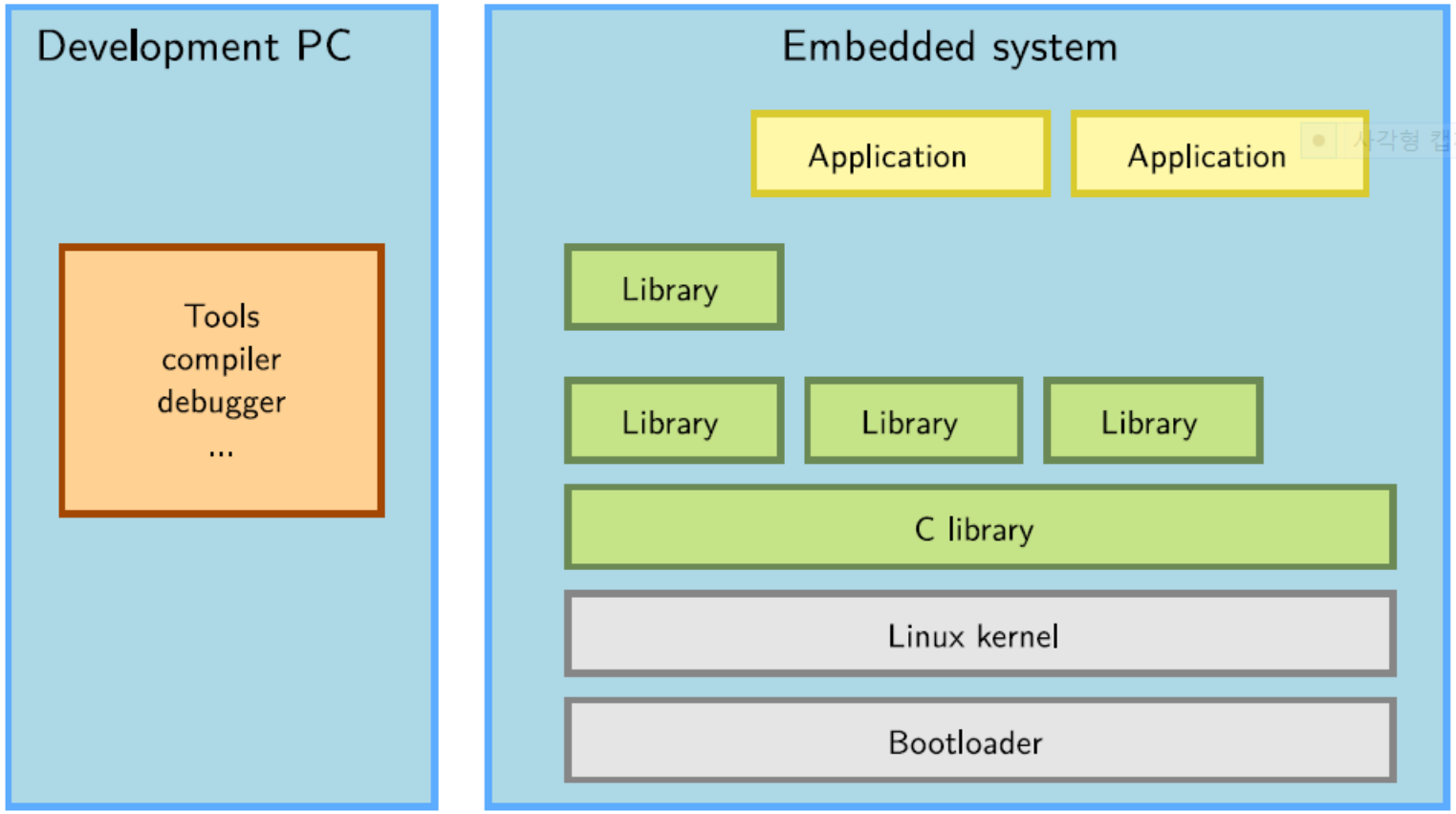
Serial Console Emulator

```
chyi@jupiter:~$ sudo picocom -b 115200 /dev/ttyUSB1
[sudo] password for chyi:
picocom v1.7

port is        : /dev/ttyUSB1
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Terminal ready

Raspbian GNU/Linux 9 michaelpi ttyAMA0
michaelpi login:
Raspbian GNU/Linux 9 michaelpi ttyAMA0
michaelpi login: [ 4388.843538] reboot: Restarting system

Raspbian GNU/Linux 9 michaelpi ttyAMA0
michaelpi login: [  916.502921] reboot: Restarting system

Raspbian GNU/Linux 9 michaelpi ttyAMA0
michaelpi login:
```

3. Embedded System(3) – Application & Kernel & Device



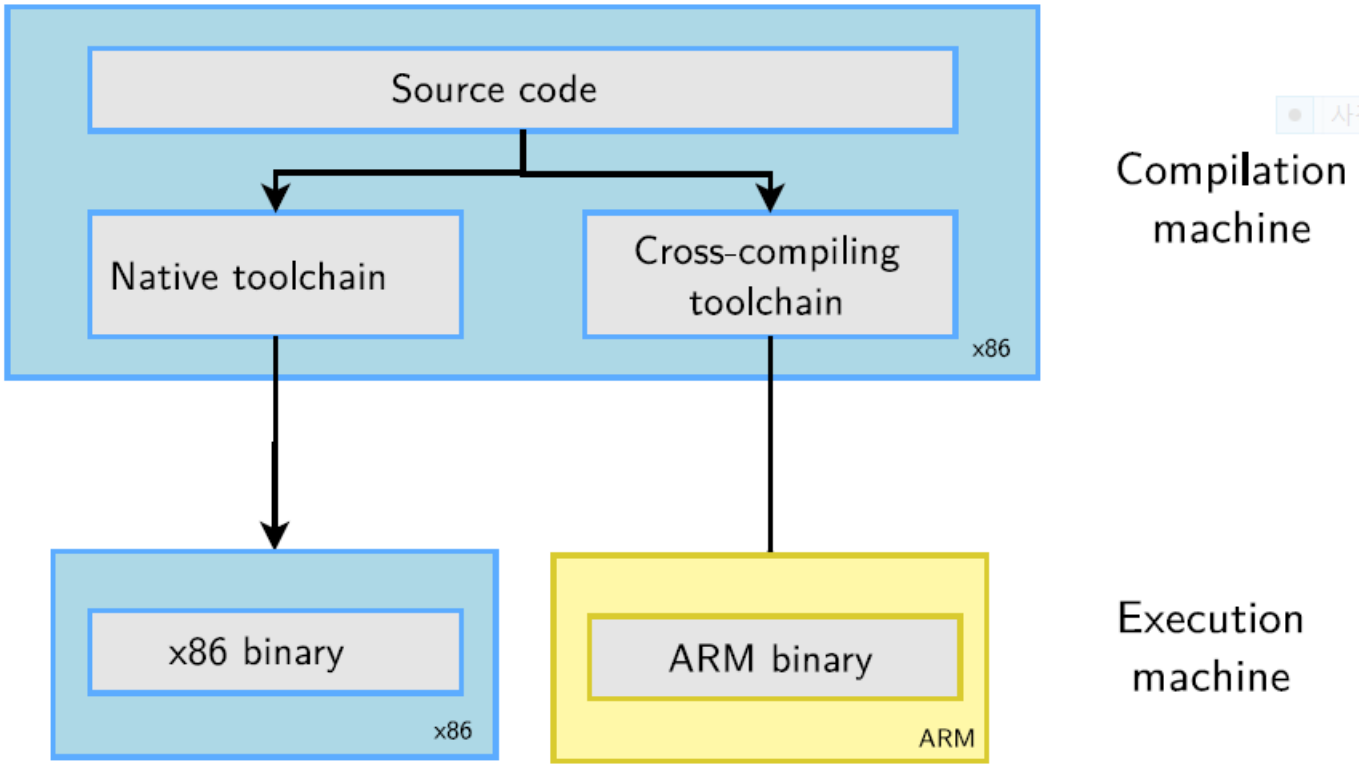
3. Embedded System(4) – S/W 구성 요소(1)



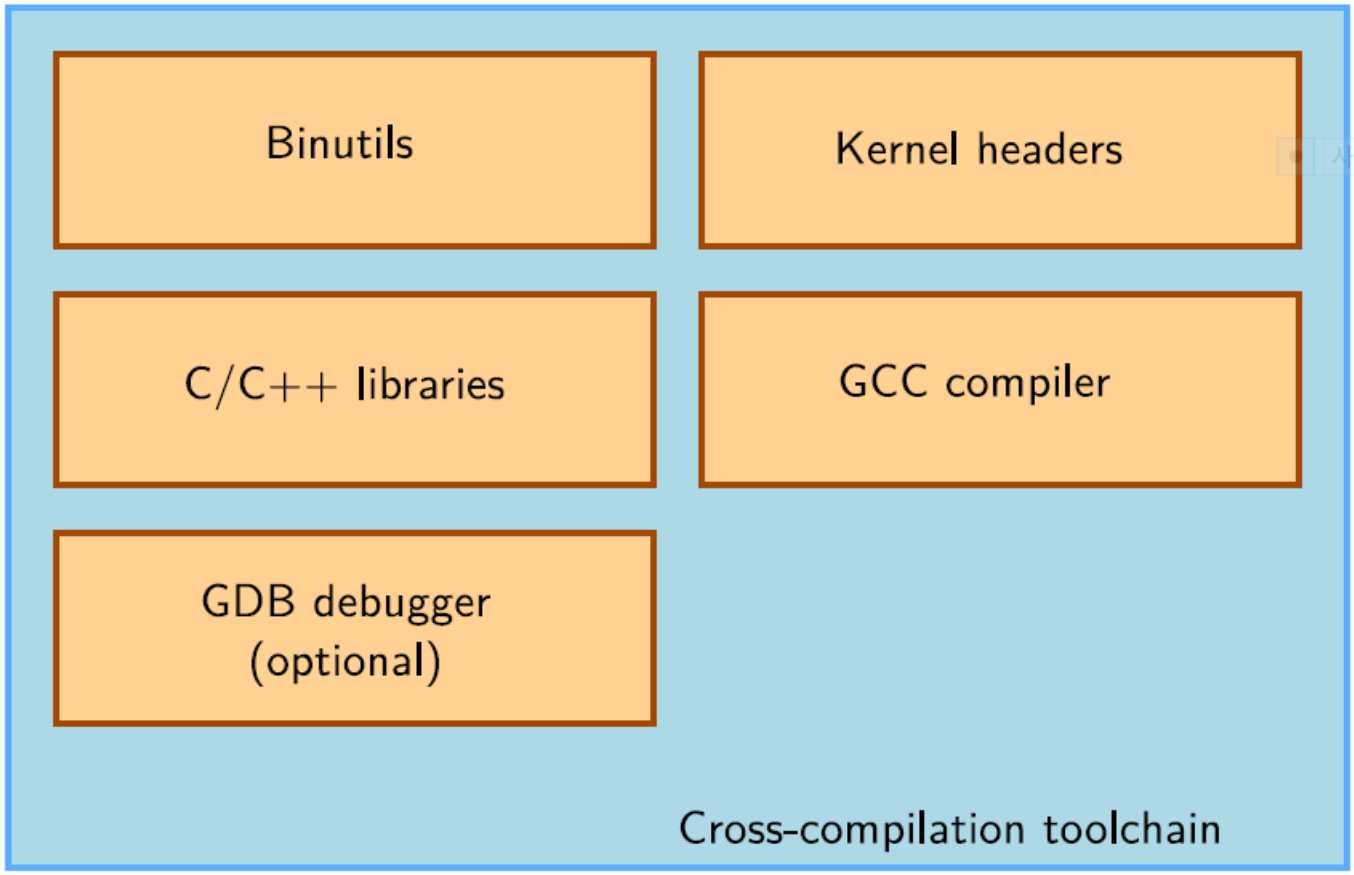
3. Embedded System(4) – S/W 구성 요소(2)

- Cross-compilation toolchain
  - Compiler that runs on the development machine, but generates code for the target
- Bootloader
  - Started by the hardware, responsible for basic initialization, loading and executing the kernel
- Linux Kernel
  - Contains the process and memory management, network stack, device drivers and provides services to user space applications
- C library
  - The interface between the kernel and the user space applications
- Libraries and applications
  - Third-party or in-house

3. Embedded System(5) – Toolchain(1)



3. Embedded System(5) – Toolchain(2)



3. Embedded System(5) – Toolchain(3) : Android

<Android ARM toolchain example>

```
chyi@jupiter:~/Android/Sdk/ndk-bundle/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64/bin$ ls
-l
합계 29792
-rwxrwxr-x 1 chyi chyi  818072  6월 12 18:22 arm-linux-androideabi-addr2line
-rwxrwxr-x 1 chyi chyi  846240  6월 12 18:22 arm-linux-androideabi-ar
-rwxrwxr-x 1 chyi chyi 1437256  6월 12 18:22 arm-linux-androideabi-as
-rwxrwxr-x 1 chyi chyi    1733  2월  5 11:11 arm-linux-androideabi-c++
-rwxrwxr-x 1 chyi chyi  816440  6월 12 18:22 arm-linux-androideabi-c++filt
-rwxrwxr-x 1 chyi chyi  789000  6월 12 18:22 arm-linux-androideabi-cpp
-rwxrwxr-x 1 chyi chyi 2913224  6월 12 18:22 arm-linux-androideabi-dwp
-rwxrwxr-x 1 chyi chyi   28584  6월 12 18:22 arm-linux-androideabi-elfedit
-rwxrwxr-x 1 chyi chyi  789000  6월 12 18:22 arm-linux-androideabi-g++
-rwxrwxr-x 1 chyi chyi  784904  6월 12 18:22 arm-linux-androideabi-gcc
-rwxrwxr-x 1 chyi chyi    1733  2월  5 11:11 arm-linux-androideabi-gcc-4.9
-rwxrwxr-x 1 chyi chyi  784904  6월 12 18:22 arm-linux-androideabi-gcc-4.9.x
-rwxrwxr-x 1 chyi chyi   25440  6월 12 18:22 arm-linux-androideabi-gcc-ar
-rwxrwxr-x 1 chyi chyi   25408  6월 12 18:22 arm-linux-androideabi-gcc-nm
-rwxrwxr-x 1 chyi chyi   25408  6월 12 18:22 arm-linux-androideabi-gcc-ranlib
-rwxrwxr-x 1 chyi chyi  426056  2월  5 11:11 arm-linux-androideabi-gcov
-rwxrwxr-x 1 chyi chyi  462984  6월 12 18:22 arm-linux-androideabi-gcov-tool
-rwxrwxr-x 1 chyi chyi  883064  6월 12 18:22 arm-linux-androideabi-gprof
-rwxrwxr-x 1 chyi chyi 5073016  6월 12 18:22 arm-linux-androideabi-ld
-rwxrwxr-x 1 chyi chyi 1318920  6월 12 18:22 arm-linux-androideabi-ld.bfd
-rwxrwxr-x 1 chyi chyi 5073016  6월 12 18:22 arm-linux-androideabi-ld.gold
-rwxrwxr-x 1 chyi chyi  829784  6월 12 18:22 arm-linux-androideabi-nm
-rwxrwxr-x 1 chyi chyi 1009336  6월 12 18:22 arm-linux-androideabi-objcopy
-rwxrwxr-x 1 chyi chyi 1288888  6월 12 18:22 arm-linux-androideabi-objdump
-rwxrwxr-x 1 chyi chyi  846240  6월 12 18:22 arm-linux-androideabi-ranlib
-rwxrwxr-x 1 chyi chyi  503568  6월 12 18:22 arm-linux-androideabi-readelf
-rwxrwxr-x 1 chyi chyi  818968  6월 12 18:22 arm-linux-androideabi-size
-rwxrwxr-x 1 chyi chyi  818392  6월 12 18:22 arm-linux-androideabi-strings
-rwxrwxr-x 1 chyi chyi 1009336  6월 12 18:22 arm-linux-androideabi-strip
chyi@jupiter:~/Android/Sdk/ndk-bundle/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64/bin$
```

3. Embedded System(5) – Toolchain(4) : RPi3(1)

$ git clone https://github.com/raspberrypi/tools ~/tools

$ **export PATH=$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin**

$ ./arm-linux-gnueabihf-gcc -v

<Raspberry Pi3 toolchain example>

```
chyi@jupiter:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin$ ls -la
합계 26720
drwxrwxr-x 2 chyi chyi    4096  7월  2 14:29 .
drwxrwxr-x 7 chyi chyi    4096  7월  2 14:29 ..
-rwxrwxr-x 1 chyi chyi  774904  7월  2 14:29 arm-linux-gnueabihf-addr2line
-rwxrwxr-x 1 chyi chyi  807608  7월  2 14:29 arm-linux-gnueabihf-ar
-rwxrwxr-x 1 chyi chyi 1372600  7월  2 14:29 arm-linux-gnueabihf-as
lrwxrwxrwx 1 chyi chyi      23  7월  2 14:29 arm-linux-gnueabihf-c++ -> arm-linux-gnueabihf-g++
-rwxrwxr-x 1 chyi chyi  774520  7월  2 14:29 arm-linux-gnueabihf-c++filt
-rwxrwxr-x 1 chyi chyi  739112  7월  2 14:29 arm-linux-gnueabihf-cpp
-rw-rw-r-- 1 chyi chyi    2981  7월  2 14:29 arm-linux-gnueabihf-ct-ng.config
-rwxrwxr-x 1 chyi chyi 2896224  7월  2 14:29 arm-linux-gnueabihf-dwp
-rwxrwxr-x 1 chyi chyi   31520  7월  2 14:29 arm-linux-gnueabihf-elfedit
-rwxrwxr-x 1 chyi chyi  739112  7월  2 14:29 arm-linux-gnueabihf-g++
lrwxrwxrwx 1 chyi chyi      29  7월  2 14:29 arm-linux-gnueabihf-gcc -> arm-linux-gnueabihf-gcc-4.8.3
-rwxrwxr-x 1 chyi chyi  739112  7월  2 14:29 arm-linux-gnueabihf-gcc-4.8.3
-rwxrwxr-x 1 chyi chyi   27040  7월  2 14:29 arm-linux-gnueabihf-gcc-ar
-rwxrwxr-x 1 chyi chyi   27040  7월  2 14:29 arm-linux-gnueabihf-gcc-nm
-rwxrwxr-x 1 chyi chyi   27040  7월  2 14:29 arm-linux-gnueabihf-gcc-ranlib
-rwxrwxr-x 1 chyi chyi  318384  7월  2 14:29 arm-linux-gnueabihf-gcov
-rwxrwxr-x 1 chyi chyi 4514808  7월  2 14:29 arm-linux-gnueabihf-gdb
-rwxrwxr-x 1 chyi chyi  739112  7월  2 14:29 arm-linux-gnueabihf-gfortran
-rwxrwxr-x 1 chyi chyi  841656  7월  2 14:29 arm-linux-gnueabihf-gprof
lrwxrwxrwx 1 chyi chyi      26  7월  2 14:29 arm-linux-gnueabihf-ld -> arm-linux-gnueabihf-ld.bfd
-rwxrwxr-x 1 chyi chyi 1259544  7월  2 14:29 arm-linux-gnueabihf-ld.bfd
-rwxrwxr-x 1 chyi chyi 3768736  7월  2 14:29 arm-linux-gnueabihf-ld.gold
-rwxrwxr-x 1 chyi chyi   10497  7월  2 14:29 arm-linux-gnueabihf-ldd
-rwxrwxr-x 1 chyi chyi  787832  7월  2 14:29 arm-linux-gnueabihf-nm
-rwxrwxr-x 1 chyi chyi  966904  7월  2 14:29 arm-linux-gnueabihf-objcopy
-rwxrwxr-x 1 chyi chyi 1196664  7월  2 14:29 arm-linux-gnueabihf-objdump
-rwxrwxr-x 1 chyi chyi     417  7월  2 14:29 arm-linux-gnueabihf-pkg-config
-rwxrwxr-x 1 chyi chyi  113560  7월  2 14:29 arm-linux-gnueabihf-pkg-config-real
-rwxrwxr-x 1 chyi chyi  807640  7월  2 14:29 arm-linux-gnueabihf-ranlib
-rwxrwxr-x 1 chyi chyi  467592  7월  2 14:29 arm-linux-gnueabihf-readelf
-rwxrwxr-x 1 chyi chyi  778904  7월  2 14:29 arm-linux-gnueabihf-size
-rwxrwxr-x 1 chyi chyi  774808  7월  2 14:29 arm-linux-gnueabihf-strings
-rwxrwxr-x 1 chyi chyi  966904  7월  2 14:29 arm-linux-gnueabihf-strip
chyi@jupiter:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin$
```

$ **arm-linux-gnueabihf-gcc** –v

$ **arm-linux-gnueabihf-gcc** –o hello hello.c

3. Embedded System(5) – Toolchain(4) : RPi3(2)

https://github.com/ChunghanYi/linux_sys_prog

$ **export ARCH=arm**
$ **export CC=arm-linux-gnueabihf-gcc**
$ **export LD=arm-linux-gnueabihf-ld**
$ **export PATH=$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin**

$ cd s_08/
$ ../genmake
$ **make**
$ **file lab1_malloc**
lab1_malloc: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, BuildID[sha1]=90dbe78cae01e2d29be20086388983cd6a08ea60, not stripped

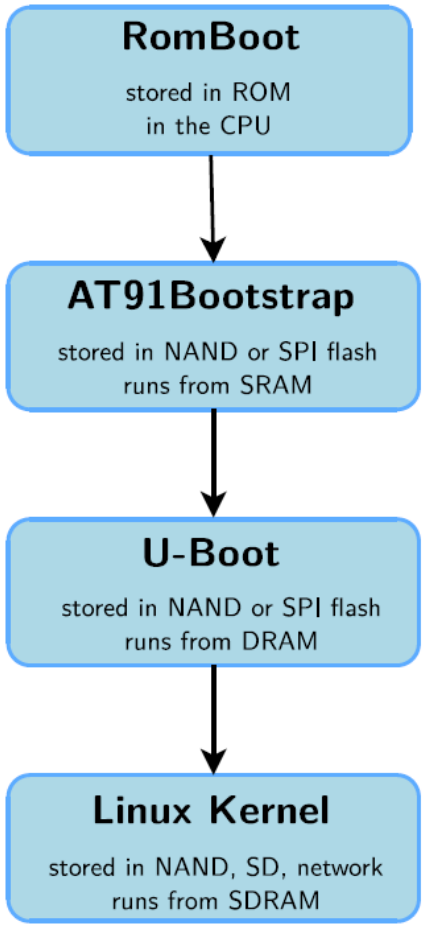
Bootloader & Device Tree (Chapter 7)

1. U-Boot Bootloader - 가장 유명한 bootloader

U-Boot is a typical free software project

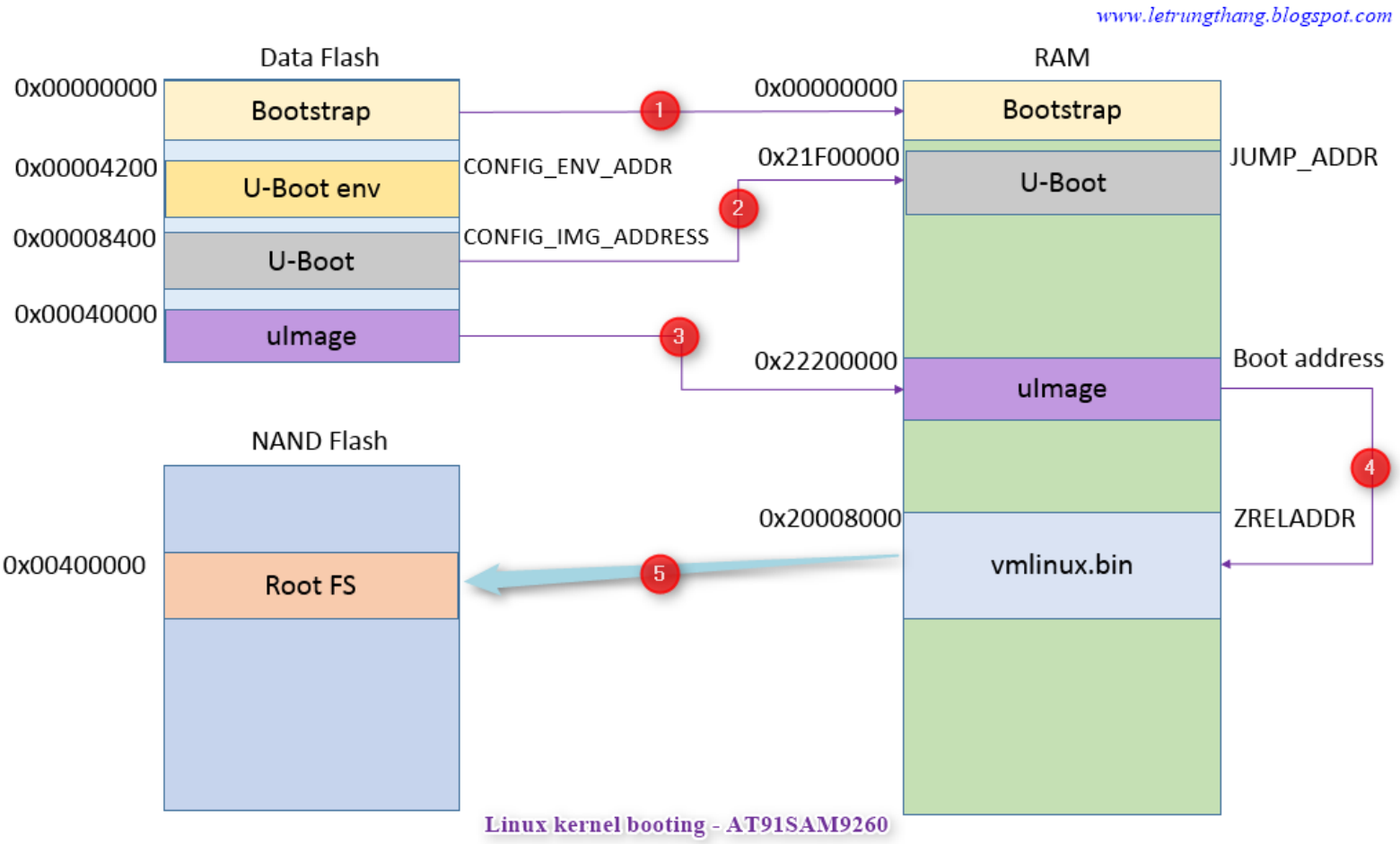
- License: GPLv2 (same as Linux)
- Freely available at http://www.denx.de/wiki/U-Boot
- Documentation available at http://www.denx.de/wiki/U-Boot/Documentation
- The latest development source code is available in a Git repository: http://git.denx.de/?p=u-boot.git;a=summary
- Development and discussions happen around an open mailing-list http://lists.denx.de/pipermail/u-boot/
- Since the end of 2008, it follows a fixed-interval release schedule. Every three months, a new version is released. Versions are named YYYY.MM.

2. Atmel AT91(1)



- **RomBoot**: tries to find a valid bootstrap image from various storage sources, and load it into SRAM (DRAM not initialized yet). Size limited to 4 KB. No user interaction possible in standard boot mode.
- **AT91Bootstrap**: runs from SRAM. Initializes the DRAM, the NAND or SPI controller, and loads the secondary bootloader into RAM and starts it. No user interaction possible.
- **U-Boot**: runs from RAM. Initializes some other hardware devices (network, USB, etc.). Loads the kernel image from storage or network to RAM and starts it. Shell with commands provided.
- **Linux Kernel**: runs from RAM. Takes over the system completely (bootloaders no longer exists).

2. Atmel AT91(2)



3. Marvell ESSPRESSObin

```
[ 5424.775282] xhci-hcd d0058000.usb3: USB bus 3 deregistered
[ 5424.780897] xhci-hcd d0058000.usb3: remove, state 4
[ 5424.785837] usb usb2: USB disconnect, device number 1
[ 5424.791163] xhci-hcd d0058000.usb3: USB bus 2 deregistered
[ 5424.797165] reboot: Restarting sy◆Booting Trusted Firmware
BL1: v1.2(release):armada-17.02.0:
BL1: Built : 09:41:56, Jun  2 2NOTICE:  BL2: v1.2(release):armada-17.02.0:
NOTICE:  BL2: Built : 09:41:57, Jun  2 20NOTICE:  BL31: v1.2(release):armada-17.02.0:
NOTICE:  BL31:

U-Boot 2015.01-armada-17.02.0-g8128e91 (Jun 02 2017 - 09:41:51)

I2C:   ready
DRAM:  1 GiB
Board: DB-88F3720-ESPRESSOBin
       CPU    @ 1000 [MHz]
       L2     @ 800 [MHz]
       TClock @ 200 [MHz]
       DDR    @ 800 [MHz]
Comphy-0: PEX0          2.5 Gbps
Comphy-1: USB3          5 Gbps
Comphy-2: SATA0         5 Gbps
Now running in RAM - U-Boot at: 3ff2b000
U-Boot DT blob at : 000000003fa18168
MMC:   XENON-SDHCI: 0
SF: Detected W25Q32DW with page size 256 Bytes, erase size 4 KiB, total 4 MiB
PCIE-0: Link down
SCSI:  SATA link 0 timeout.
AHCI 0001.0300 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: ncq led only pmp fbss pio slum part sxs
Net:   neta0
Hit any key to stop autoboot:  0
Marvell>>
Marvell>> ?
?       - alias for 'help'
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootelf - Boot from an ELF image in memory
booti   - boot arm64 Linux Image image from memory
bootm   - boot application image from memory
```

4. U-Boot 명령 입력 모드

Flash information (NOR and SPI flash)

```
U-Boot> flinfo
DataFlash:AT45DB021
Nb pages: 1024
Page Size: 264
Size= 270336 bytes
Logical address: 0xC0000000
Area 0: C0000000 to C0001FFF (RO) Bootstrap
Area 1: C0002000 to C0003FFF Environment
Area 2: C0004000 to C0041FFF (RO) U-Boot
```

NAND flash information

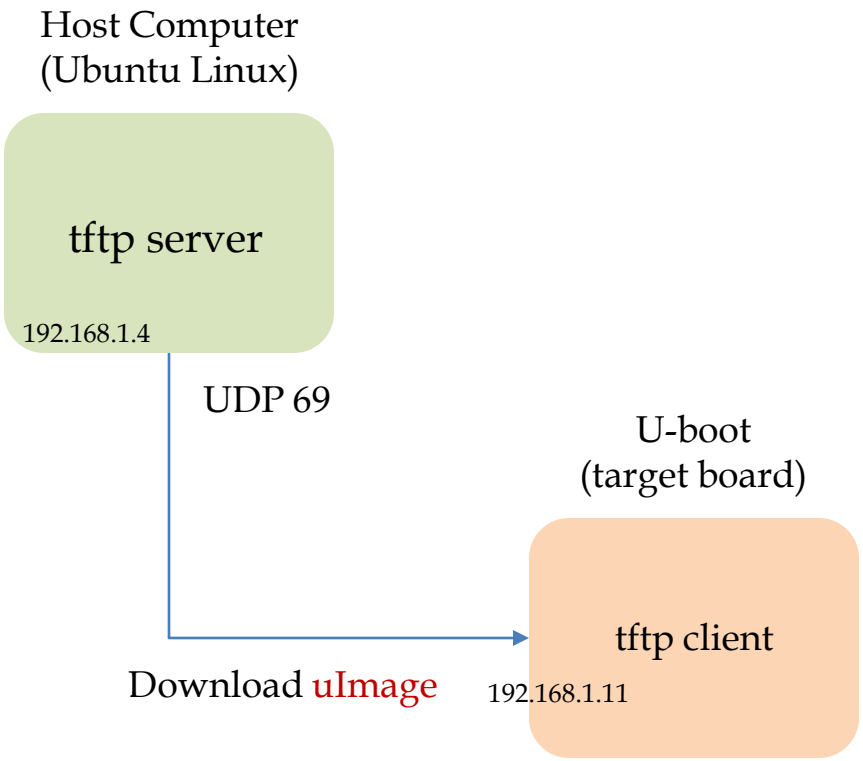
```
U-Boot> nand info
Device 0: nand0, sector size 128 KiB
  Page size       2048 b
  OOB size          64 b
  Erase size    131072 b
```

Version details

```
U-Boot> version
U-Boot 2013.04 (May 29 2013 - 10:30:21)
```

```
u-boot # printenv
baudrate=19200
ethaddr=00:40:95:36:35:33
netmask=255.255.255.0
ipaddr=10.0.0.11
serverip=10.0.0.1
stdin=serial
stdout=serial
stderr=serial
u-boot # printenv serverip
serverip=10.0.0.1
u-boot # setenv serverip 10.0.0.100
u-boot # saveenv
```

5. Kernel Image Loading at U-Boot




```
U-Boot# dhcp
link up on port 0, speed 100, full duplex
BOOTP broadcast 1
DHCP client bound to address 192.168.1.11

_____________________________________

Next will be configuring the server-ip [our host machine's IP], kernel's
command line and the load address
_____________________________________

U-Boot# setenv serverip 192.168.1.4
U-Boot# setenv bootfile uImage
U-Boot# setenv bootargs console=ttyO0,115200
root=/dev/mmcblk0p2 rw rootwait ip=dhcp
U-Boot# tftp 0x80200000 uImage
link up on port 0, speed 100, full duplex
Using cpsw device
TFTP from server 192.168.1.4; our IP address is
192.168.1.11
Filename 'uImage'.
Load address: 0x80200000
Loading:
#################################################################
#################################################################T
############
#########################################T
######################
#################################T ########
146.5 KiB/s
done
Bytes transferred = 3484264 (352a68 hex)
U-Boot# bootm 0x80200000
## Booting kernel from Legacy Image at 80200000 ...
Image Name: Angstrom/3.2.28/beaglebone
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 3484200 Bytes = 3.3 MiB
Load Address: 80008000
Entry Point: 80008000
Verifying Checksum ... OK
Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Linux version 3.2.28 (koen@Angstrom-F16-vm-
rpm) (gcc version 4.5.4 20120305 (prerelease) (GCC) ) #1
Tue Sep 11 13:08:30 CEST 2012
[ 0.000000] CPU: ARMv7 Processor [413fc082] revision 2
(ARMv7), cr=50c53c7d
[ 0.000000] CPU: PIPT / VIPT nonaliasing data cache,
VIPT aliasing instruction cache
[ 0.000000] Machine: am335xevm
```

6. U-Boot Source Download & Build 절차

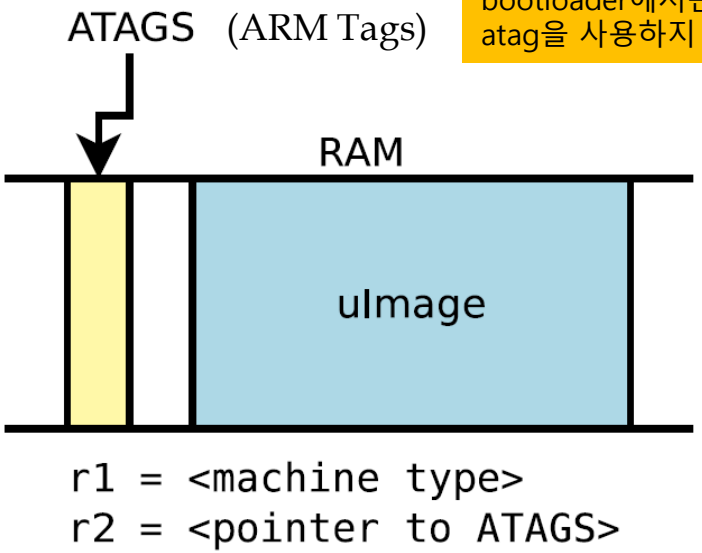
Marvell ESSPRESSObin board용 u-boot

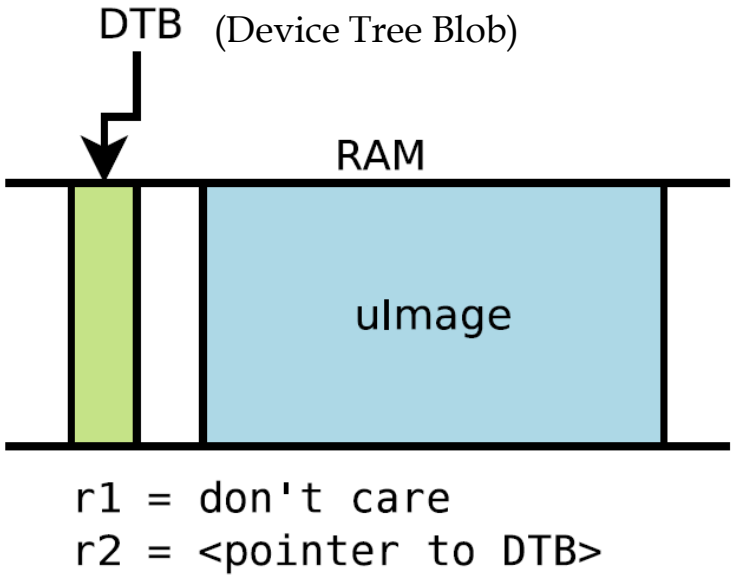
$ git clone https://github.com/MarvellEmbeddedProcessors/u-boot-marvell .
$ git checkout u-boot-2017.03-armada-17.06
$ **export CROSS_COMPILE=aarch64-linux-gnu-**
$ make **mvebu_espressobin-88f3720_defconfig**
$ make **DEVICE_TREE=armada-3720-espressobin**

$ ls -l u-boot*
-rwxrwxr-x 1 chyi chyi 3918672 7월 2 16:01 u-boot
-rw-rw-r-- 1 chyi chyi 553058 7월 2 16:01 u-boot-dtb.bin
-rw-rw-r-- 1 chyi chyi 547648 7월 2 16:01 u-boot-nodtb.bin
-rw-rw-r-- 1 chyi chyi 553058 7월 2 16:01 **u-boot.bin**
-rw-rw-r-- 1 chyi chyi 10438 7월 2 16:00 u-boot.cfg
-rw-rw-r-- 1 chyi chyi 5674 7월 2 16:01 u-boot.cfg.configs
-rw-rw-r-- 1 chyi chyi 5410 7월 2 16:01 u-boot.dtb
-rw-rw-r-- 1 chyi chyi 1304 7월 2 16:01 u-boot.lds
-rw-rw-r-- 1 chyi chyi 432664 7월 2 16:01 u-boot.map
-rw-rw-r-- 1 chyi chyi 1574550 7월 2 16:01 u-boot.srec
-rw-rw-r-- 1 chyi chyi 192149 7월 2 16:01 u-boot.sym

7. Device Tree(1) - ATAGS(OLD) & DTB(NEW)

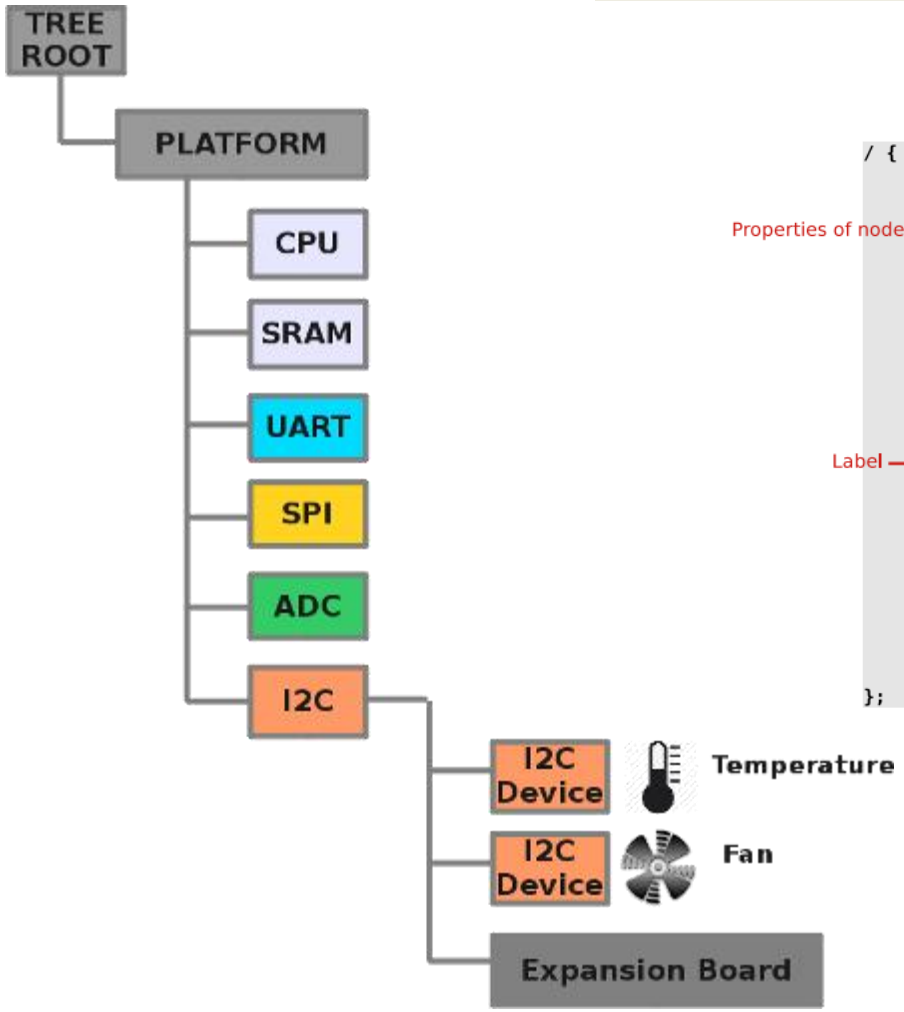
atag은 bootloader에서 linux kernel로 설정 정보를 전달하기 위해 사용함.
bootloader에서는 atag에 대한 포인터를 kernel 호출인자로 반드시 넘겨줘야 함.
atag을 사용하지 않을 경우 0으로 초기화 함.

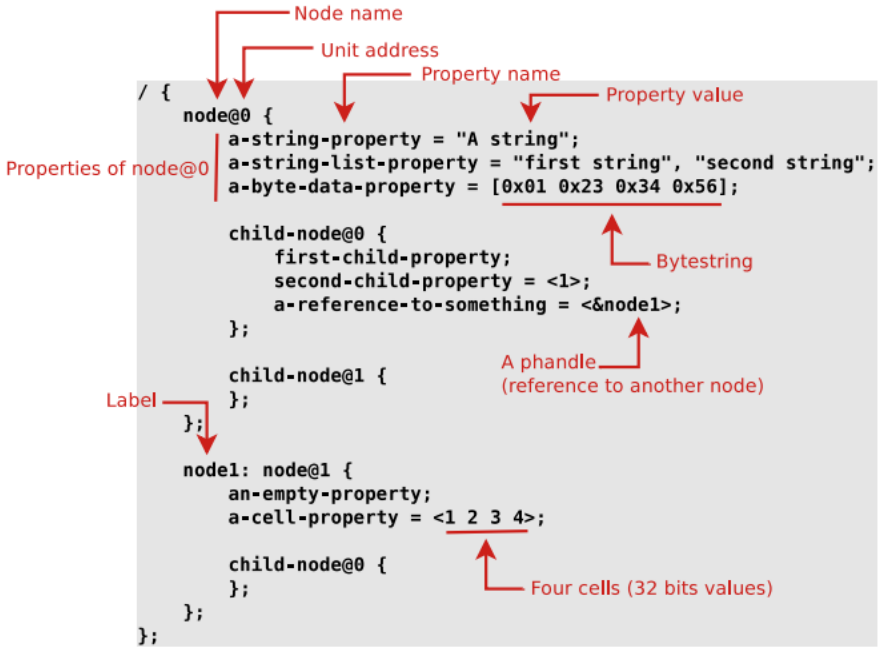




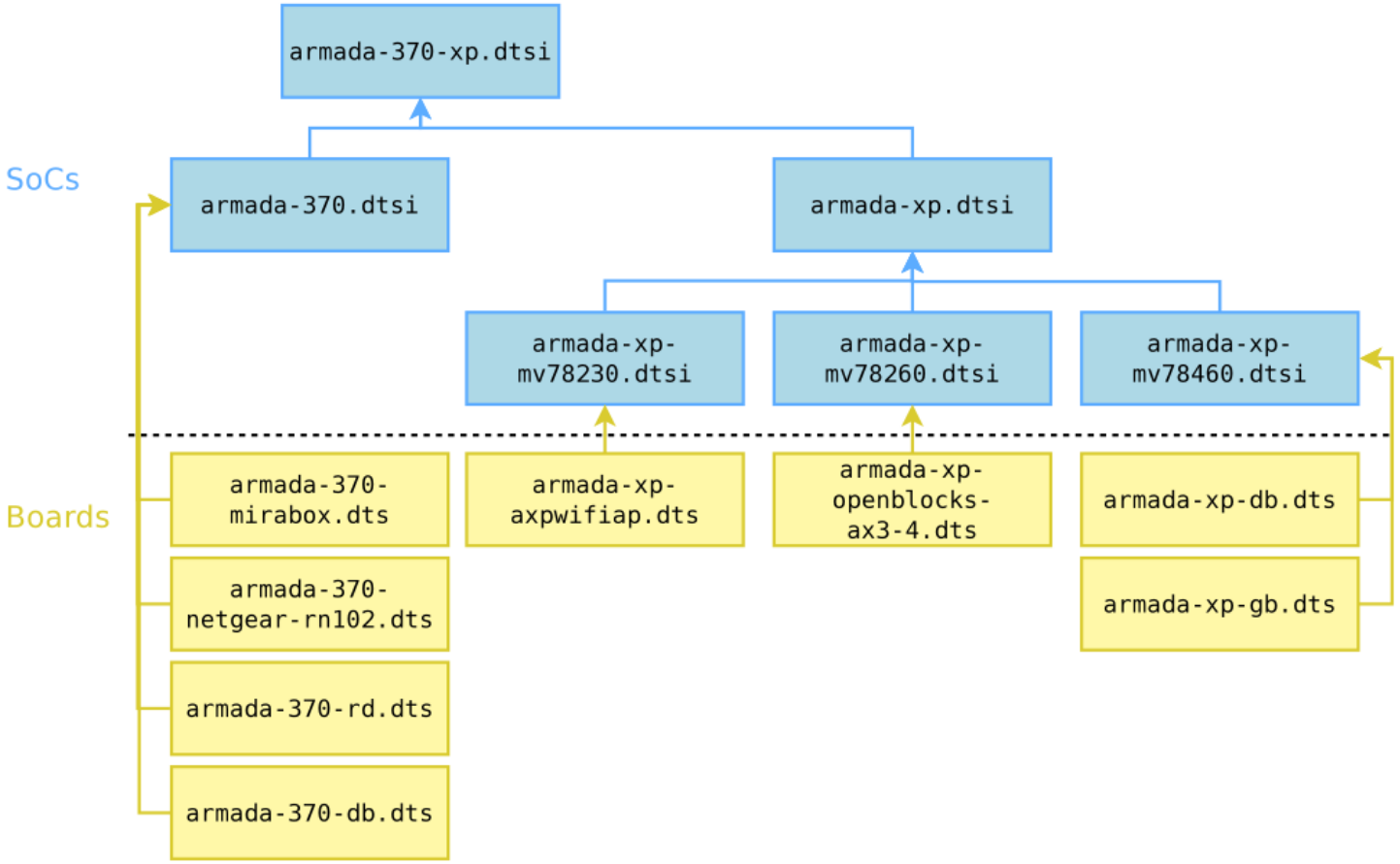
7. Device Tree(2)

<Device Tree 관련 참조 Site>
http://slowbootkernelhacks.blogspot.com/2014/03/beaglebone-linux-kernel310x-programming.html





7. Device Tree(3)



Device Tree Compiler(DTC)로 Device Tree Source(DTS)를 compile(DTB)하여 사용함.

7. Device Tree(4) – RPi3 Device Tree Example

```
#include "bcm2837.dtsi"
#include "bcm270x.dtsi"
#include "bcm2708-rpi.dtsi"

/ {
    compatible = "brcm,bcm2837", "brcm,bcm2836";

    soc {

        arm-pmu {
#ifdef RPI364
            compatible = "arm,armv8-pmuv3", "arm,cortex-a7-pmu";
#else
            compatible = "arm,cortex-a7-pmu";
#endif
            interrupt-parent = <&local_intc>;
            interrupts = <9>;
        };

        /delete-node/ timer@7e003000;
    };

    __overrides__ {
        arm_freq = <&cpu0>, "clock-frequency:0",
               <&cpu1>, "clock-frequency:0",
               <&cpu2>, "clock-frequency:0",
               <&cpu3>, "clock-frequency:0";
    };
};
```

<arch/arm/boot/dts/bcm2710.dtsi>

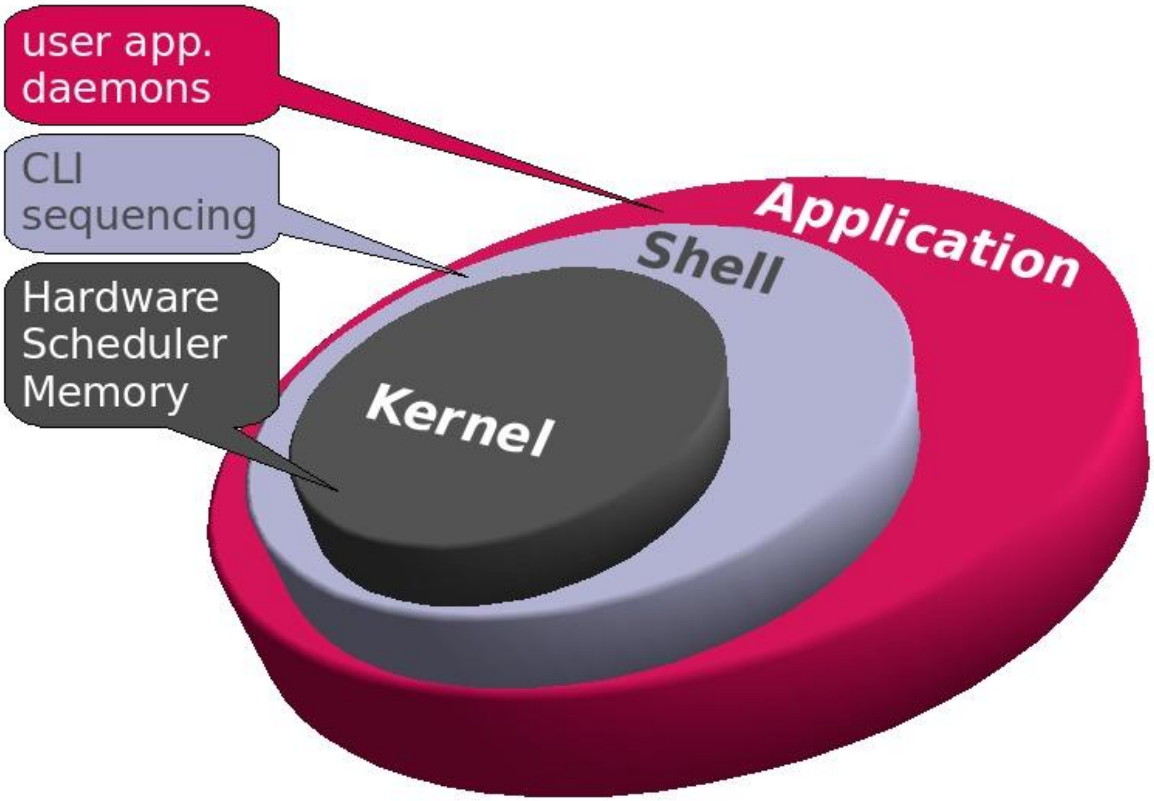
```
/dts-v1/;

#include "bcm2710.dtsi"
#include "bcm283x-rpi-smsc9514.dtsi"

/ {
    compatible = "raspberrypi,3-model-b", "brcm,bcm2837";
    model = "Raspberry Pi 3 Model B";

    chosen {
        bootargs = "8250.nr_uarts=1";
    };

    aliases {
        serial0 = &uart1;
        serial1 = &uart0;
    };
};

&gpio {
    spi0_pins: spi0_pins {
        brcm,pins = <9 10 11>;
        brcm,function = <4>; /* alt0 */
    };

    spi0_cs_pins: spi0_cs_pins {
        brcm,pins = <8 7>;
        brcm,function = <1>; /* output */
    };

    i2c0_pins: i2c0 {
        brcm,pins = <0 1>;
        brcm,function = <4>;
    };

    i2c1_pins: i2c1 {
        brcm,pins = <2 3>;
        brcm,function = <4>;
    };

    i2s_pins: i2s {
        brcm,pins = <18 19 20 21>;
        brcm,function = <4>; /* alt0 */
    };

    sdio_pins: sdio_pins {
```

<arch/arm/boot/dts/bcm2710-rpi-3-b.dts>

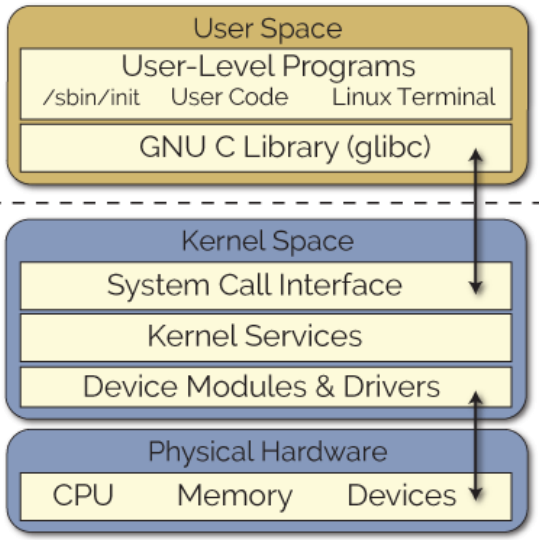
Chapter 4-5.

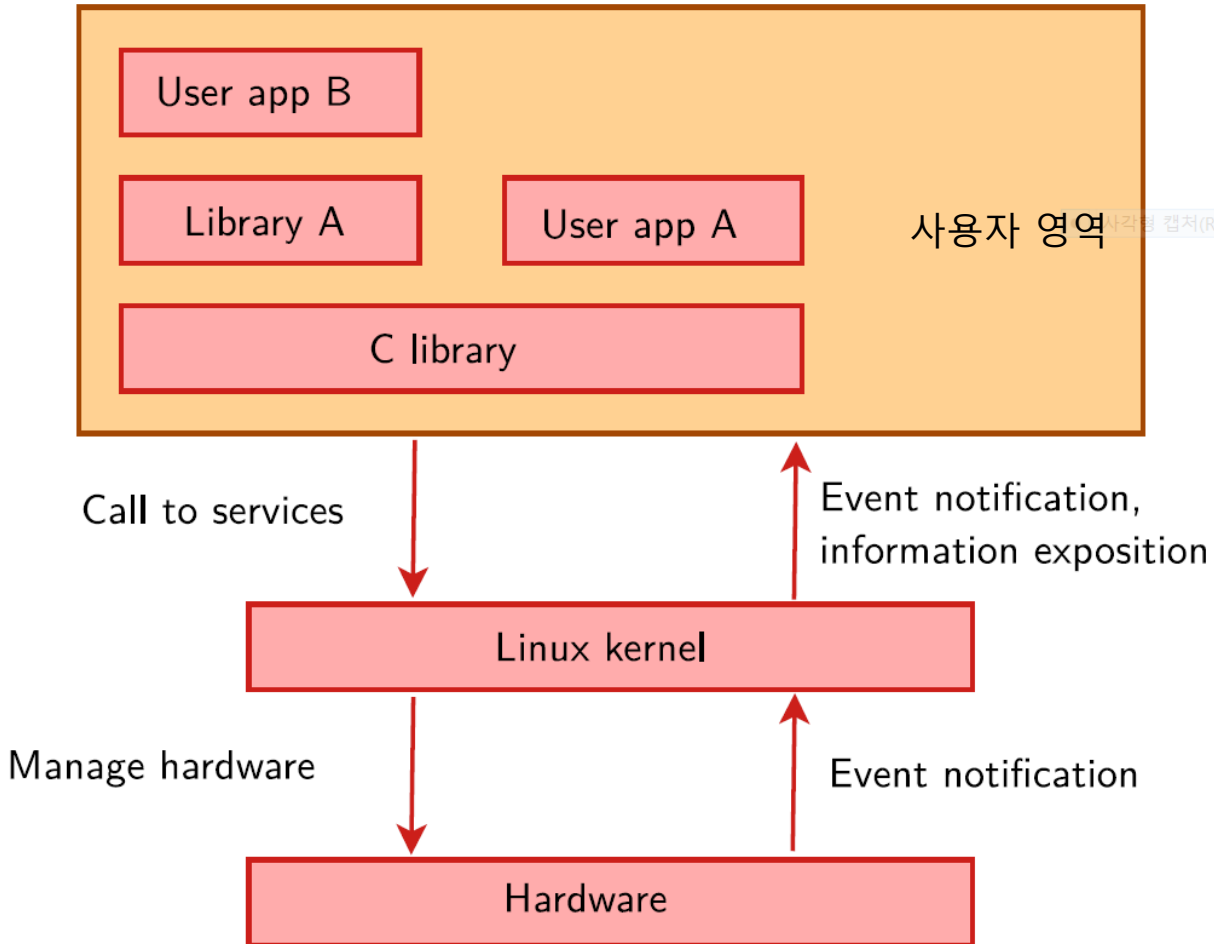
: kernel build system & kernel 초기화 과정

1. Linux Kernel 개요(1)

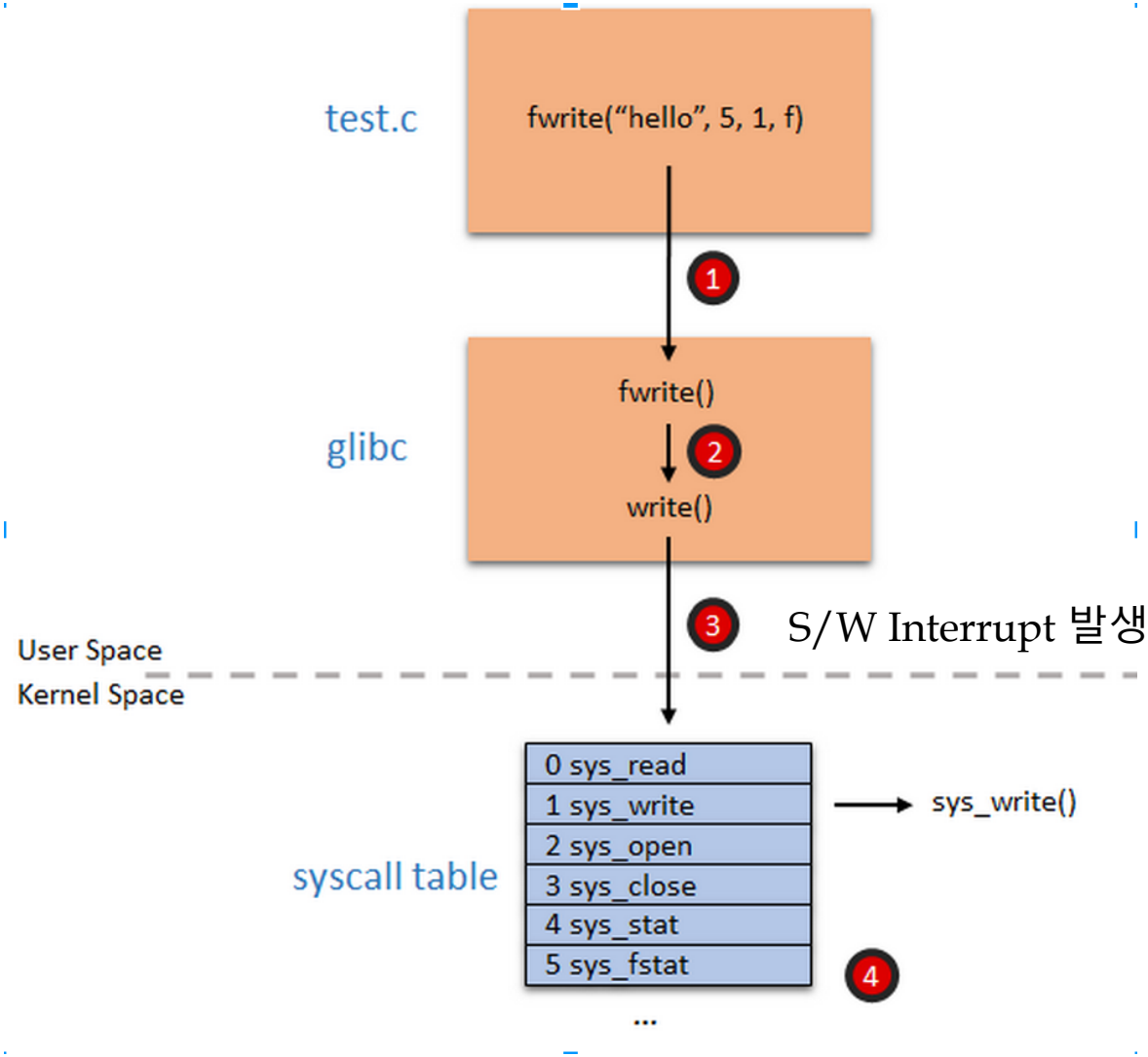


1. Linux Kernel 개요(2)

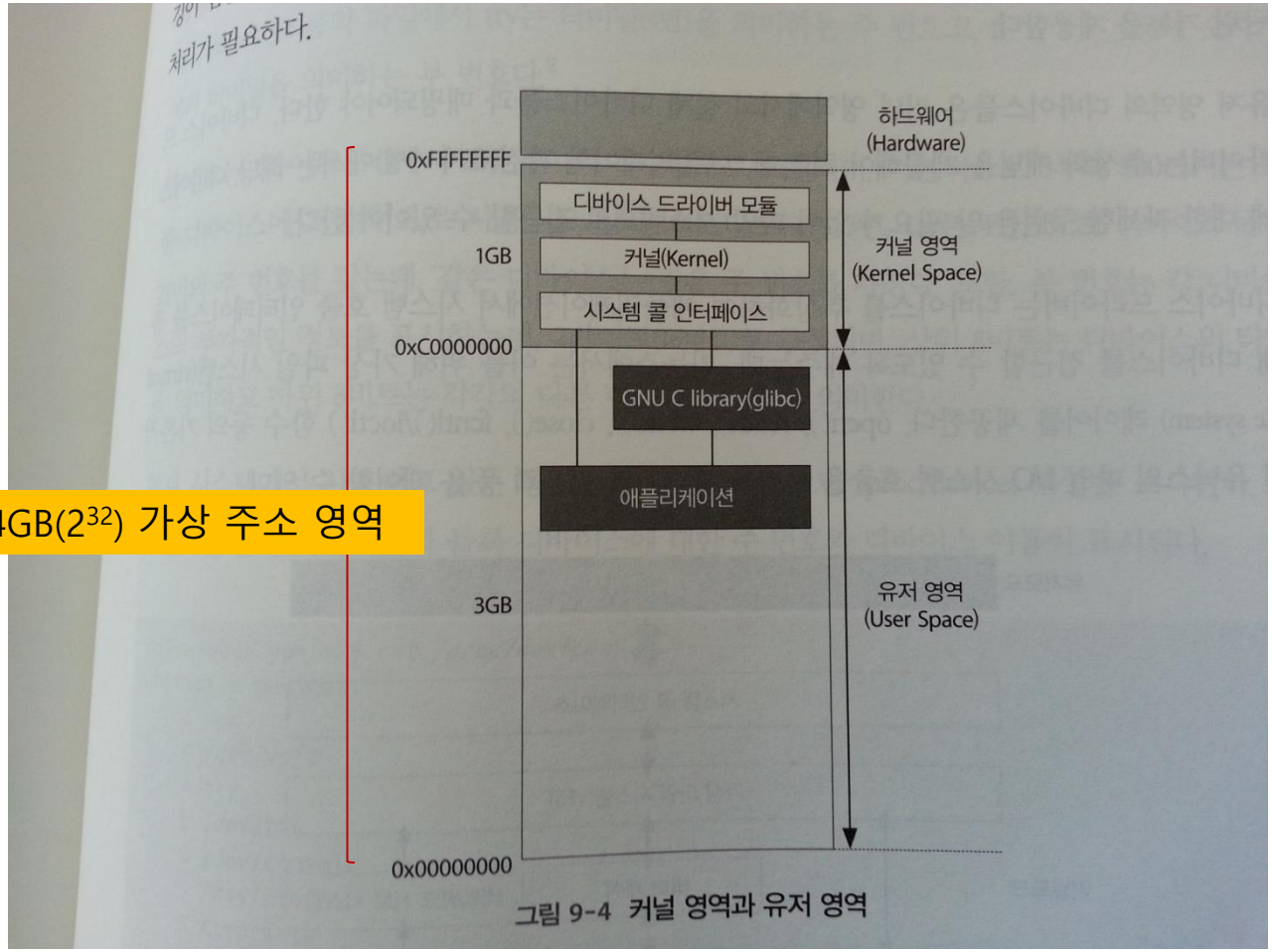




1. Linux Kernel 개요(3) – System call Interrupt



1. Linux Kernel 개요(4) – kernel 영역과 사용자 영역(1)



1. Linux Kernel 개요(4) – kernel 영역과 사용자 영역(2)

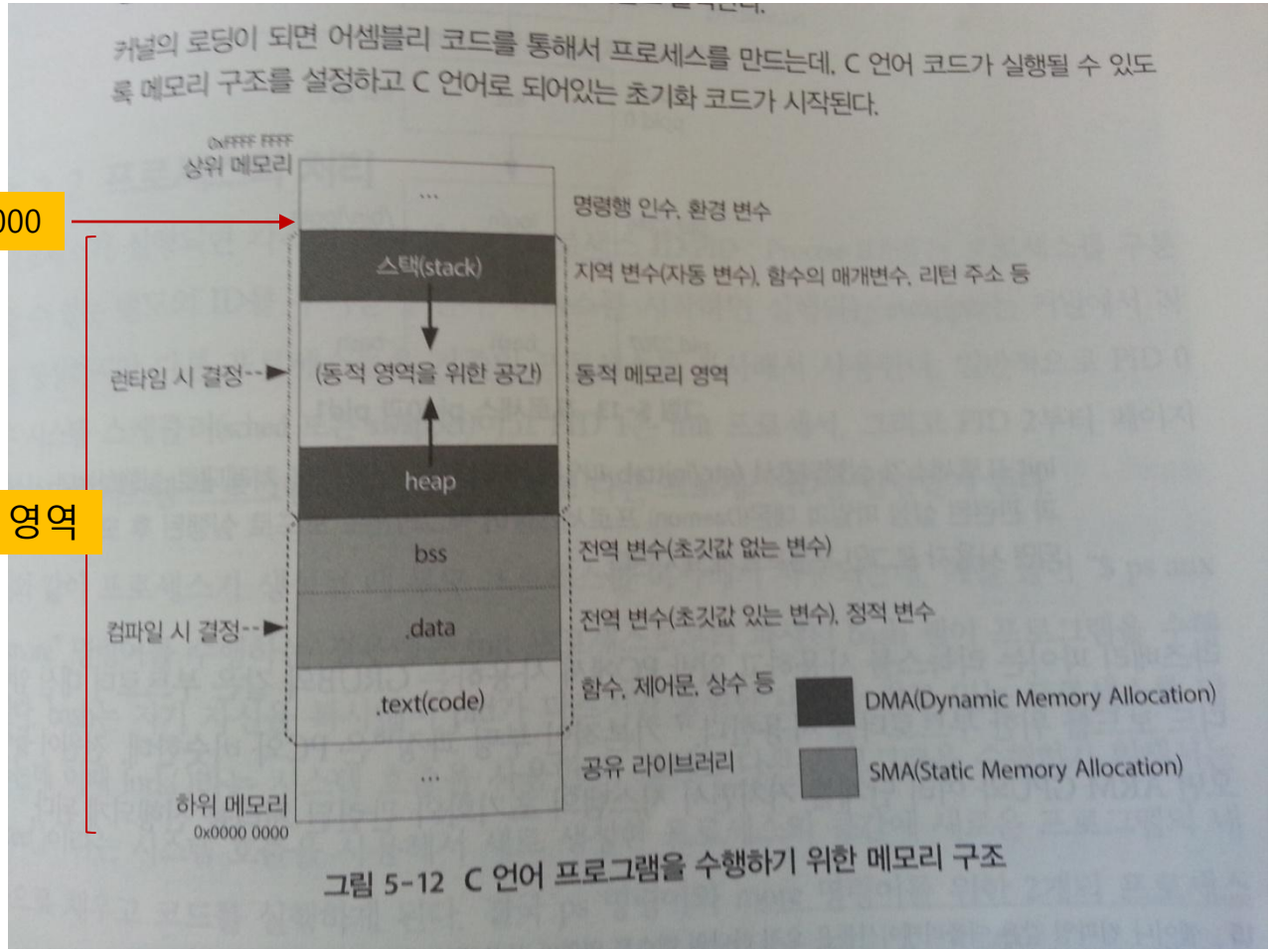


그림 5-12 C 언어 프로그램을 수행하기 위한 메모리 구조

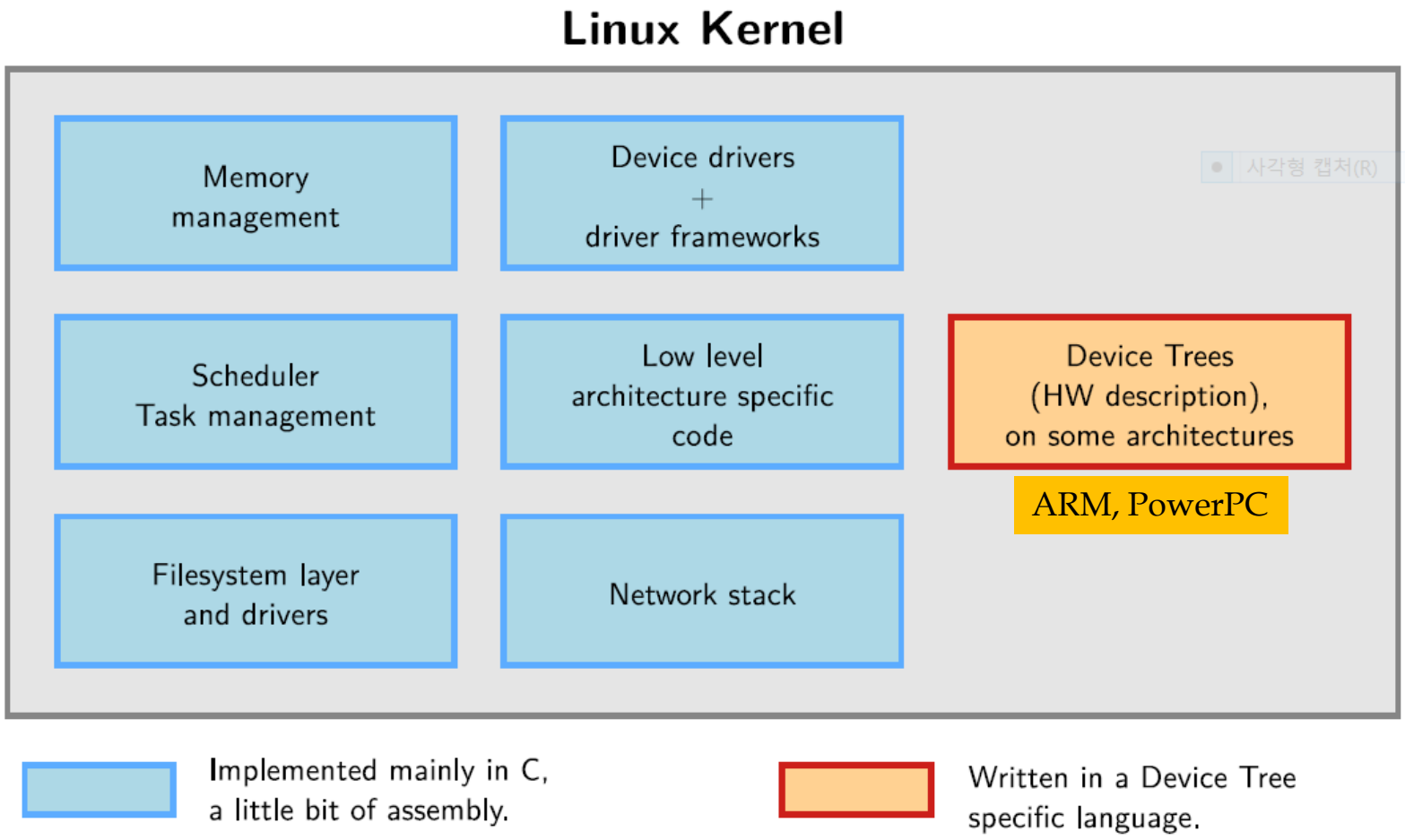
1. Linux Kernel 개요(5) – 디렉토리 구성

https://www.kernel.org/

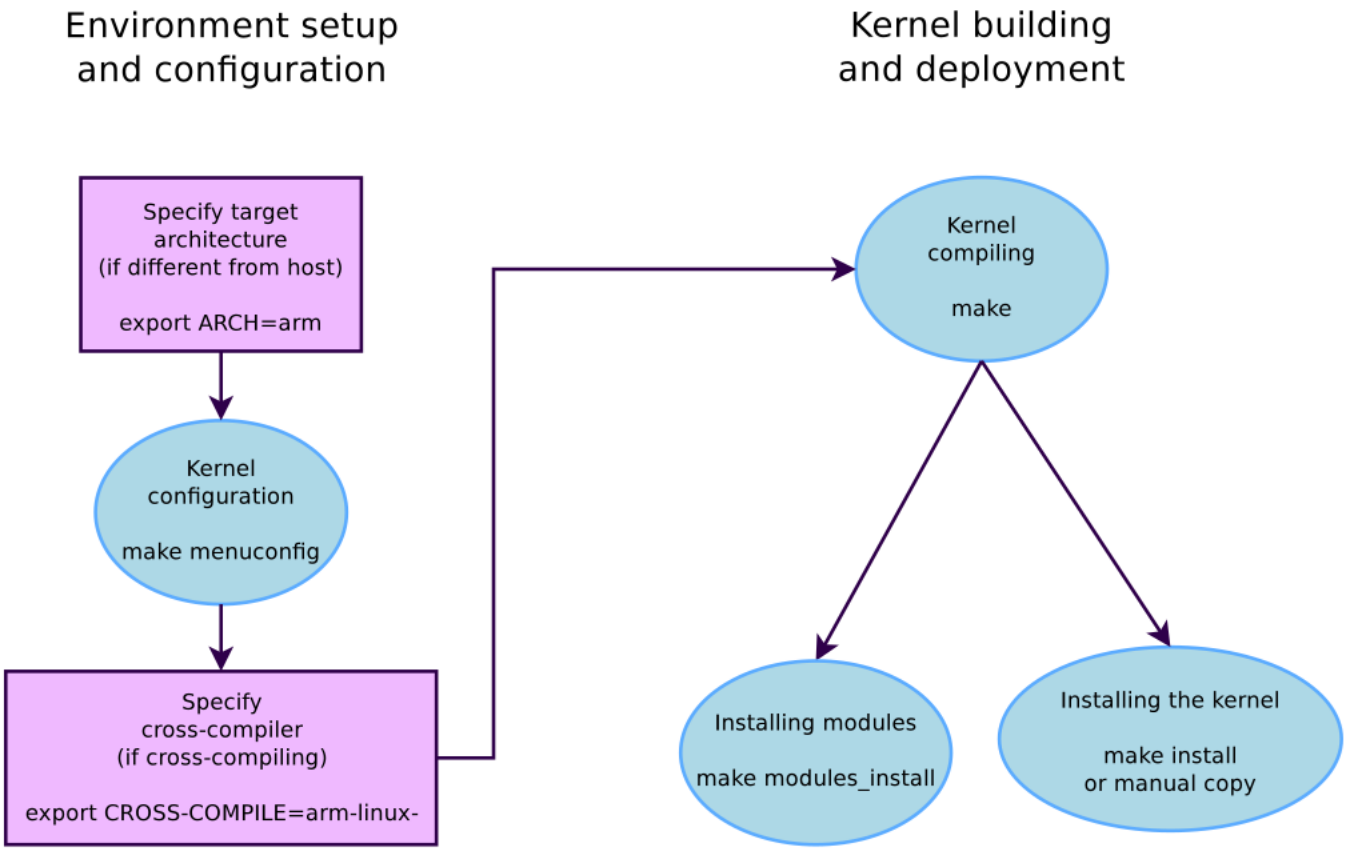
- drivers/: 49.4%
- arch/: 21.9%
- fs/: 6.0%
- include/: 4.7%
- sound/: 4.4%
- Documentation/: 4.0%
- net/: 3.9%
- firmware/: 1.0%
- kernel/: 1.0%
- tools/: 0.9%
- scripts/: 0.5%
- mm/: 0.5%
- crypto/: 0.4%
- security/: 0.4%
- lib/: 0.4%
- block/: 0.2%
- ...

1. Linux Kernel 개요(5)

(*) Device Tree 관련해서는 참고 문헌 [4] 참조



1. Linux Kernel 개요(6) – build 절차(1)



1. Linux Kernel 개요(6) – build 절차(2)

make mrproper	커널 설정 초기화(*항상 수행하는 것 아님*)
make YOURBOARD_defconfig	Target board에 맞는 config 설정 (arch/arm/configs 디렉토리에 있음) => .config 파일 생성됨.
make menuconfig	Kernel configuration 조정 => .config의 내용이 변경됨.
make zImage	압축된 kernel image 생성 => zImage, bzImage, uImage 등 종류마다 다름.
make modules make modules_install	Kernel 모듈 생성 및 설치(rootfs 디렉토리에)
make dtbs	DTB 생성(*ARM, PowerPPC 등에서만 필요*)

2. Linux Kernel Source Download & Build – RPi3

See https://www.raspberrypi.org/documentation/linux/kernel/building.md

$ git clone https://github.com/raspberrypi/tools ~/tools

$ echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin

$ git clone --depth=1 https://github.com/raspberrypi/linux

$ KERNEL=kernel7
$ **make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig**

$ **make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig**

$ **make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs**

<kernel module 복사하기>
$ **sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=mnt/ext4 modules_install**

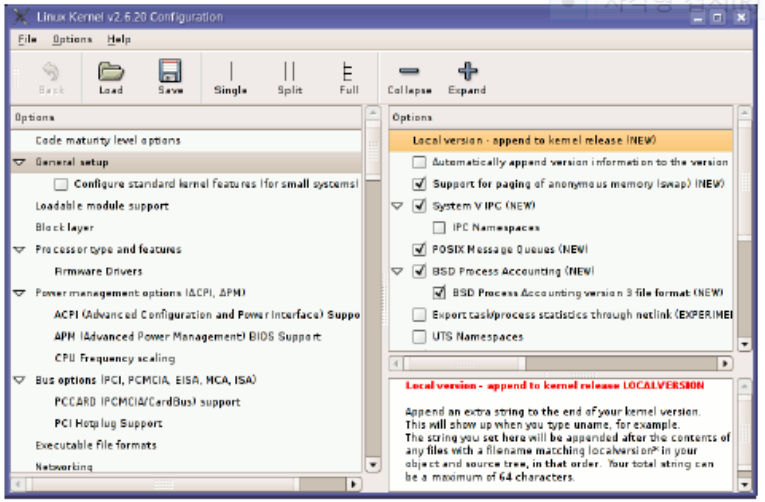
3. Kernel Build System(1) - .config

```
#
# CD-ROM/DVD Filesystems
#
CONFIG_ISO9660_FS=m
CONFIG_JOLIET=y
CONFIG_ZISOFS=y
CONFIG_UDF_FS=y
CONFIG_UDF_NLS=y

#
# DOS/FAT/NT Filesystems
#
# CONFIG_MSDOS_FS is not set
# CONFIG_VFAT_FS is not set
CONFIG_NTFS_FS=m
# CONFIG_NTFS_DEBUG is not set
CONFIG_NTFS_RW=y
```

3. Kernel Build System(2) – make gconfig

`make gconfig`

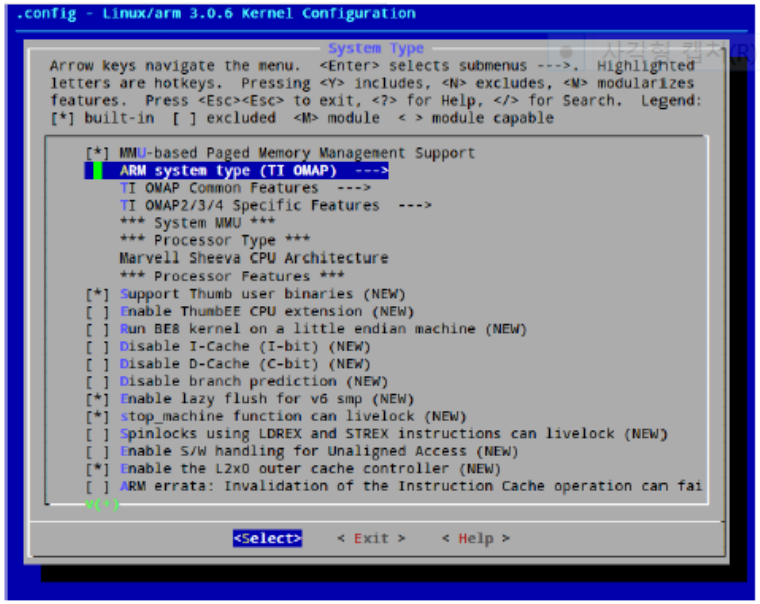
- *GTK* based graphical configuration interface. Functionality similar to that of make `xconfig`.
- Just lacking a search functionality.
- Required Debian packages: `libglade2-dev`



3. Kernel Build System(3) – make menuconfig

`make menuconfig`

- Useful when no graphics are available. Pretty convenient too!
- Same interface found in other tools: BusyBox, Buildroot...
- Required Debian packages: `libncurses-dev`



(*) 내용을 살펴 보자.

3. Kernel Build System(4) – Kconfig & Makefile

LISTING 4-8 Snippet from .../arch/arm/Kconfig

```
source "init/Kconfig"

menu "System Type"

choice
        prompt "ARM system type"
        default ARCH_RPC

config ARCH_CLPS7500
        bool "Cirrus-CL-PS7500FE"

config ARCH_CLPS711X
        bool "CLPS711x/EP721x-based"

...

source "arch/arm/mach-ixp4xx/Kconfig"
```

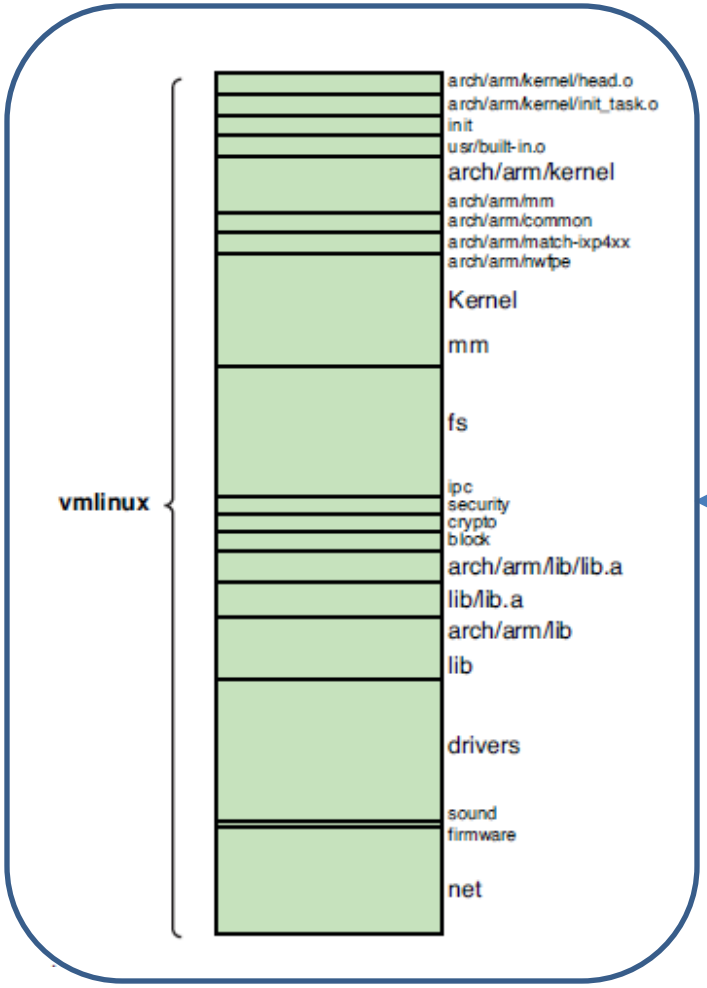
LISTING 4-11 Makefile from .../arch/arm/mach-ixp4xx Kernel Subdirectory

```
#
# Makefile for the linux kernel.
#

obj-y   += common.o common-pci.o

obj-$(CONFIG_ARCH_IXDP4XX)    += ixdp425-pci.o ixdp425-setup.o
obj-$(CONFIG_MACH_IXDPG425)    += ixdpg425-pci.o coyote-setup.o
obj-$(CONFIG_ARCH_ADI_COYOTE)  += coyote-pci.o coyote-setup.o
obj-$(CONFIG_MACH_GTWX5715)    += gtwx5715-pci.o gtwx5715-setup.o
```

4. Kernel Build 결과물(1)

```
$ make ARCH=arm CROSS_COMPILE=xscale_be- zImage
  CHK     include/linux/version.h
  UPD     include/linux/version.h
  Generating include/asm-arm/mach-types.h
  CHK     include/linux/utsrelease.h
  UPD     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-arm
```

[7] Executable and Linking Format, a de facto standard format for binary executable files.



일반적인 kernel build 과정

4.2

LISTING 4-1 Continued

```
  CC      kernel/bounds.s
  GEN     include/linux/bounds.h
  CC      arch/arm/kernel/asm-offsets.s
  .
  . <hundreds of lines of output omitted here>
  .
  LD      vmlinux
  SYSMAP  System.map
  SYSMAP  .tmp_System.map
  OBJCOPY arch/arm/boot/Image                    <-- [1]
  Kernel: arch/arm/boot/Image is ready
  AS      arch/arm/boot/compressed/head.o
  GZIP    arch/arm/boot/compressed/piggy.gz
  AS      arch/arm/boot/compressed/piggy.o
  CC      arch/arm/boot/compressed/misc.o
  AS      arch/arm/boot/compressed/head-xscale.o
  AS      arch/arm/boot/compressed/big-endian.o
  LD      arch/arm/boot/compressed/vmlinux       <-- [2]
  OBJCOPY arch/arm/boot/zImage                   <-- [3]
  Kernel: arch/arm/boot/zImage is ready
```

4. Kernel Build 결과물(2) - Kernel Image 구성

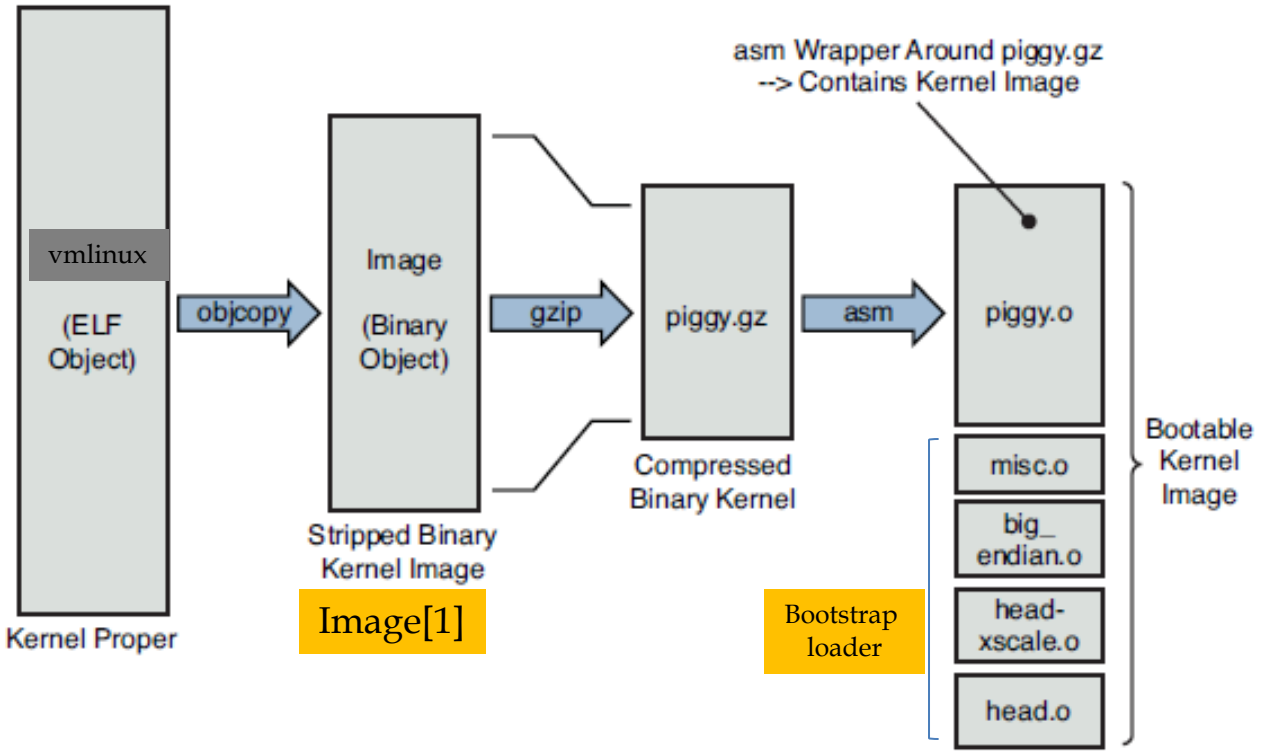


FIGURE 5-1 Composite kernel image construction

5. Kernel 초기화 과정(1)

커널 초기화 과정을 요약해 보면 다음과 같다.

1) bootloader는 bootstrap code를 실행시킨다.

2) Bootstrap 코드는 프로세서와 보드를 초기화하고, 커널의 압축을 풀어 RAM에 적재한 후, start_kernel() 함수를 호출해 준다.

3) 커널은 bootloader로부터 command line option을 복사해 온다.

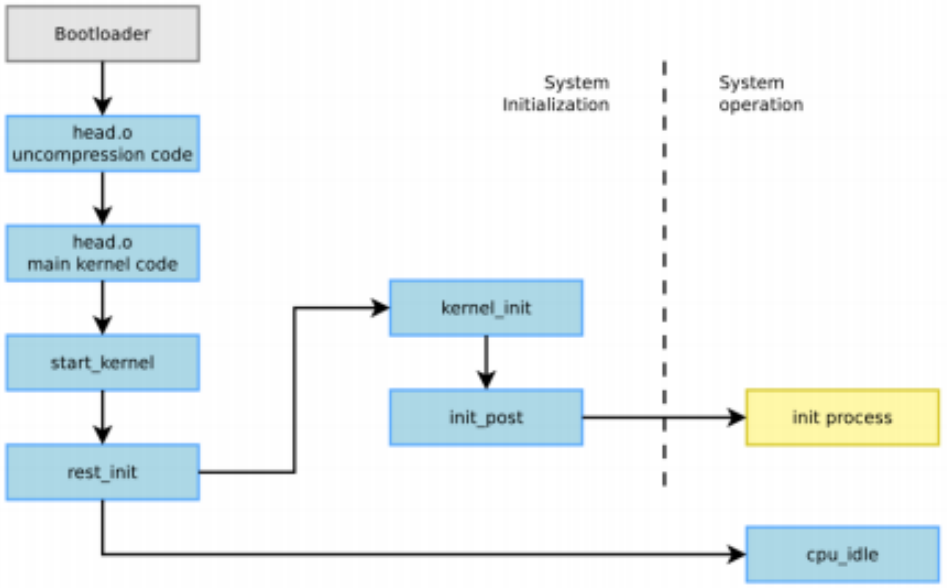
4) 커널은 프로세서와 머신을 초기화시킨다.

5) 콘솔을 초기화한다.

6) 메모리 할당(memory allocation), scheduling, 파일 cache 등 커널 서비스를 초기화 시킨다.

7) 커널 thread(나중에 init process)를 생성하고, idle loop 상태에서 대기한다.

8) 장치를 초기화하고, initcall 매크로를 호출한다.



5. Kernel 초기화 과정(2) – start_kernel

아키텍쳐 특화(architecture-specific)된 초기화 코드

커널이 스스로 압축을 푼 후, 커널의 시작 부분(kernel entry point)로 분기하게 되는데, 이를 담고 있는 파일은 arch/<arch>/kernel/head.S 이며, 주요 임무는 다음과 같다.

1) architecture, 프로세서, 머신 유형 등을 검사한다.

Check the architecture, processor and machine type.

2) MMU를 설정하고, page table 항목을 만든 후, 가상 메모리(virtual memory)를 enable 시킨다.

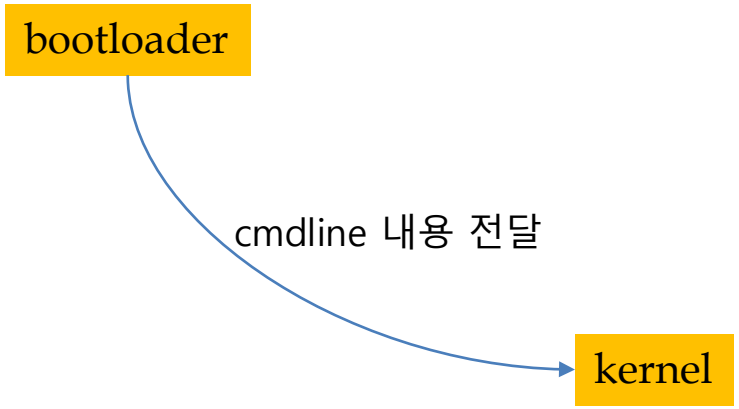
3) init/main.c 파일에 있는 start_kernel() 함수를 호출한다.

start kernel 함수에 주로 하는 일

1) setup_arch(&command_line) 함수를 호출해 준다.
   - arch/<arch>/kernel/setup.c 파일에 정의되어 있는 함수이다.
   - Bootloader가 넘긴 command line 값을 복사한다.
   - ARM에서는 이 함수는 setup_processor() 함수(CPU 정보가 출력됨)와 setup_machine() 함수(머신을 초기화 시켜줌)를 다시 호출한다
2) 에러 메시지를 출력할 수 있도록, 가능한 한 일찍 콘솔을 초기화해 준다.
3) 다양한 커널 subsystem을 초기화 시켜준다.
4) 최종적으로 rest_init() 함수를 호출한다.

5. Kernel 초기화 과정(3) - Command Line

dwc_otg.lpm_enable=0 **console=ttyAMA0,115200** console=tty1 **root=/dev/mmcblk0p2** **rootfstype=ext4** elevator=deadline rootwait



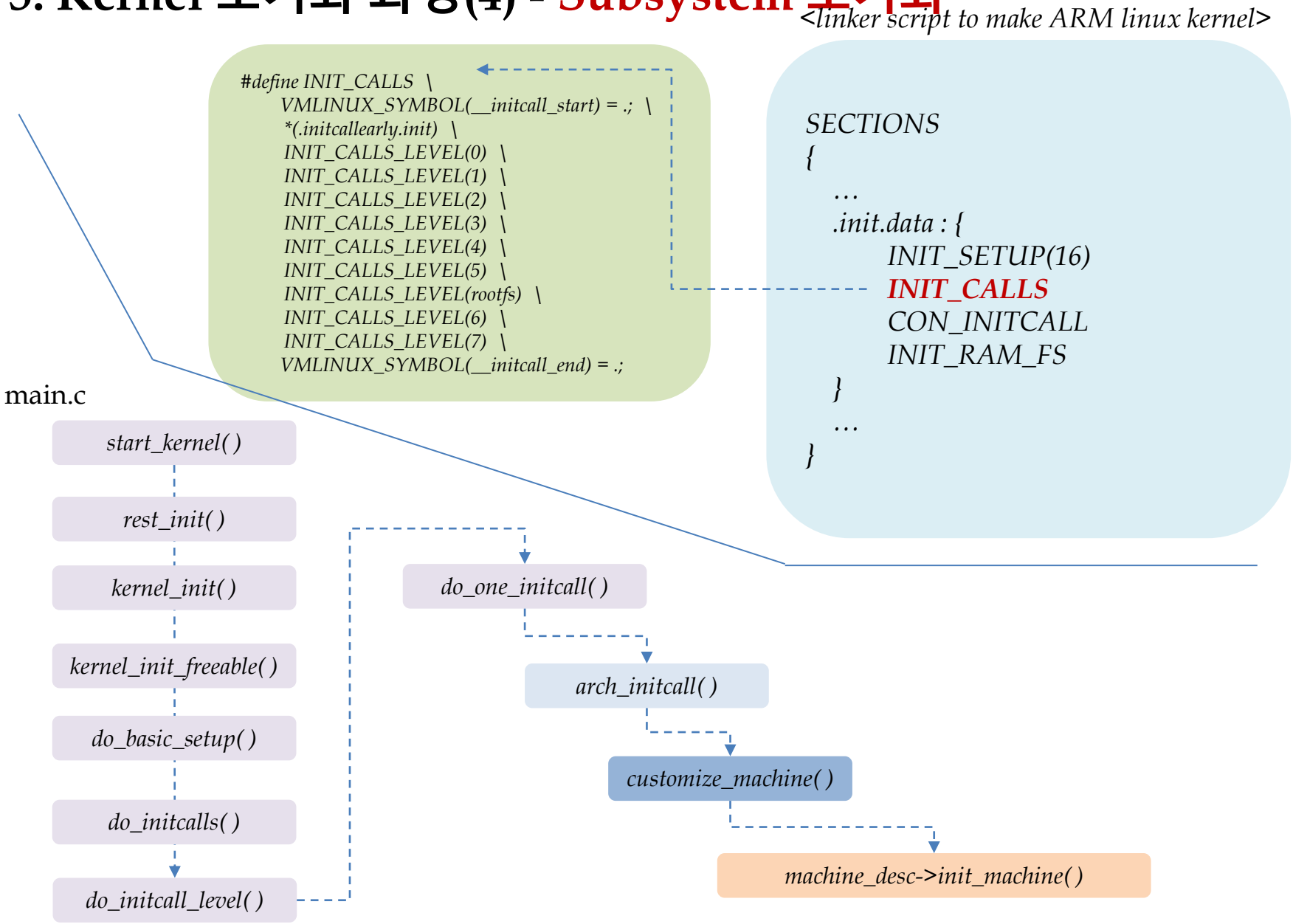
LISTING 5-4 Console Setup Code Snippet

```
/*
 *    Setup a list of consoles. Called from init/main.c
 */
static int __init console_setup(char *str)
{
    char buf[sizeof(console_cmdline[0].name) + 4]; /* 4 for index */
    char *s, *options, *brl_options = NULL;
    int idx;

    ...
    <body omitted for clarity...>
    ...
```

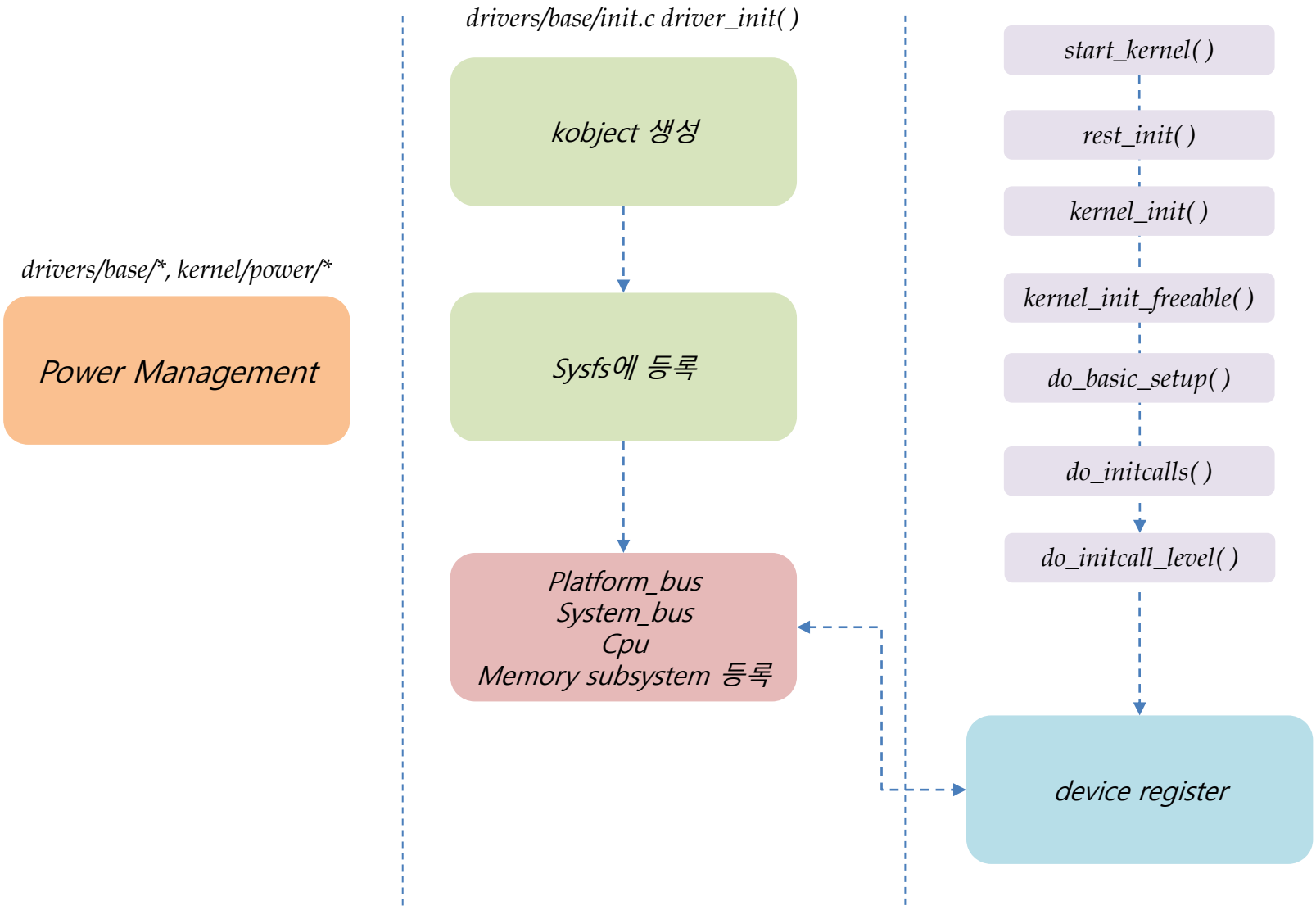

LISTING 5-4 Continued

```
    return 1;
}

__setup("console=", console_setup);
```

5. Kernel 초기화 과정(4) - Subsystem 초기화



5. Kernel 초기화 과정(5) - 디바이스 드라이버 초기화



5. Kernel 초기화 과정(6) - Init process 실행(마지막 step)

LISTING 5-11 Final Kernel Boot Steps from `main.c`

```
static noinline int init_post(void)
    __releases(kernel_lock)
{
<... lines trimmed for clarity ...>
...
if (execute_command) {
      run_init_process(execute_command);
      printk(KERN_WARNING "Failed to execute %s.  Attempting "
                        "defaults...\n", execute_command);
}

run_init_process("/sbin/init");
run_init_process("/etc/init");
run_init_process("/bin/init");
run_init_process("/bin/sh");

panic("No init found.  Try passing init= option to kernel.");
```

(*) ARM linux kernel의 boot flow 관련하여 자세한 정보를 알고 싶으면, 참고 문헌 [5]를 참고하기 바람.

6. Kernel Module Programming(1)

```c
/* hello.c */
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

static int __init hello_init(void)
{
  pr_alert("Good morrow to this fair assembly.\n");
  return 0;
}

static void __exit hello_exit(void)
{
  pr_alert("Alas, poor world, what treasure hast thou lost!\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Greeting module");
MODULE_AUTHOR("William Shakespeare");
```

6. Kernel Module Programming(2)

- The below `Makefile` should be reusable for any single-file out-of-tree Linux module
- The source file is `hello.c`
- Just run `make` to build the `hello.ko` file

```
ifneq ($(KERNELRELEASE),)
obj-m := hello.o
else
KDIR := /path/to/kernel/sources

all:
<tab>$(MAKE) -C $(KDIR) M=$$PWD
endif
```

6. Kernel Module Programming(3)

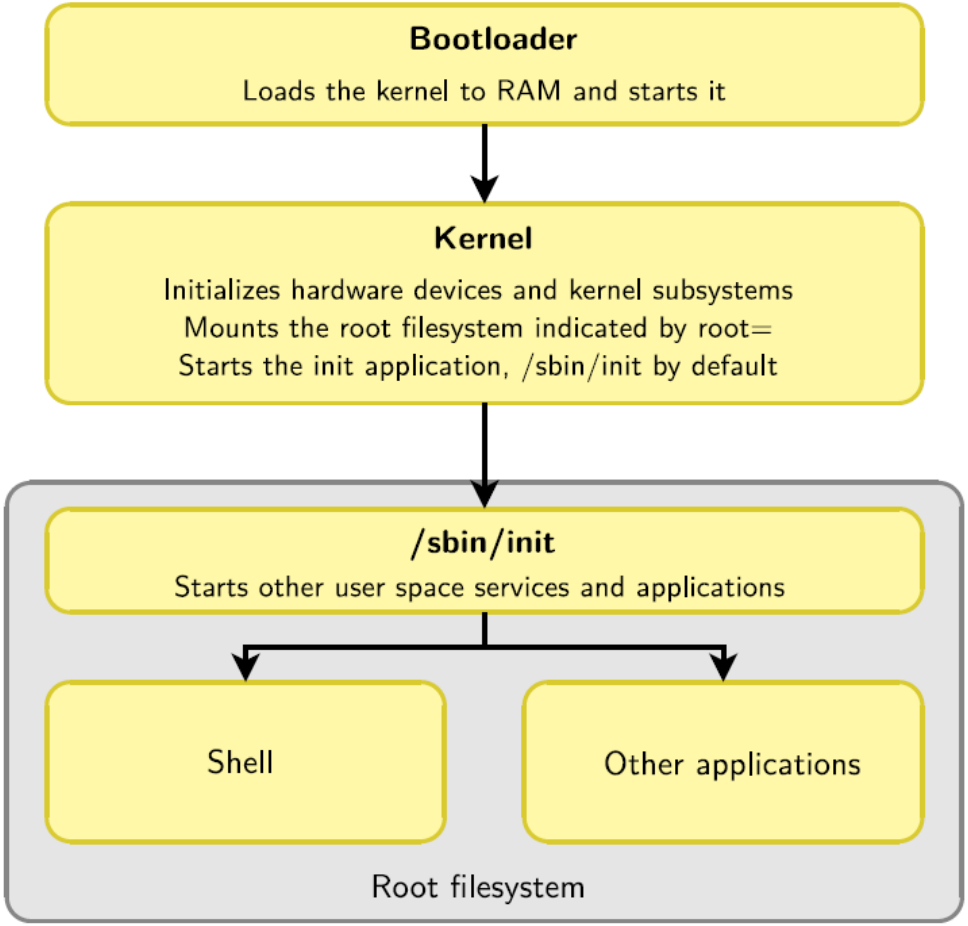
- http://slowbootkernelhacks.blogspot.com/2017/03/linux-kernel-programming-guide.html
- <예제 코드>
- **https://github.com/ChunghanYi/linux_kernel_hacks/tree/master/writing_device_drivers_by_coopj**

```
<How to build with buildroot ARM toolchain>
============================================
$ cd s_02

$ export KROOT=$YOUR_PATH/buildroot/output/build/linux-4.9
$ export ARCH=arm
$ export CROSS_COMPILE=arm-linux-
$ export CC=arm-linux-gcc
$ export LD=arm-linux-ld
$ export PATH=$YOUR_PATH/buildroot/output/host/usr/bin:$PATH

$ ../genmake

$ make
```

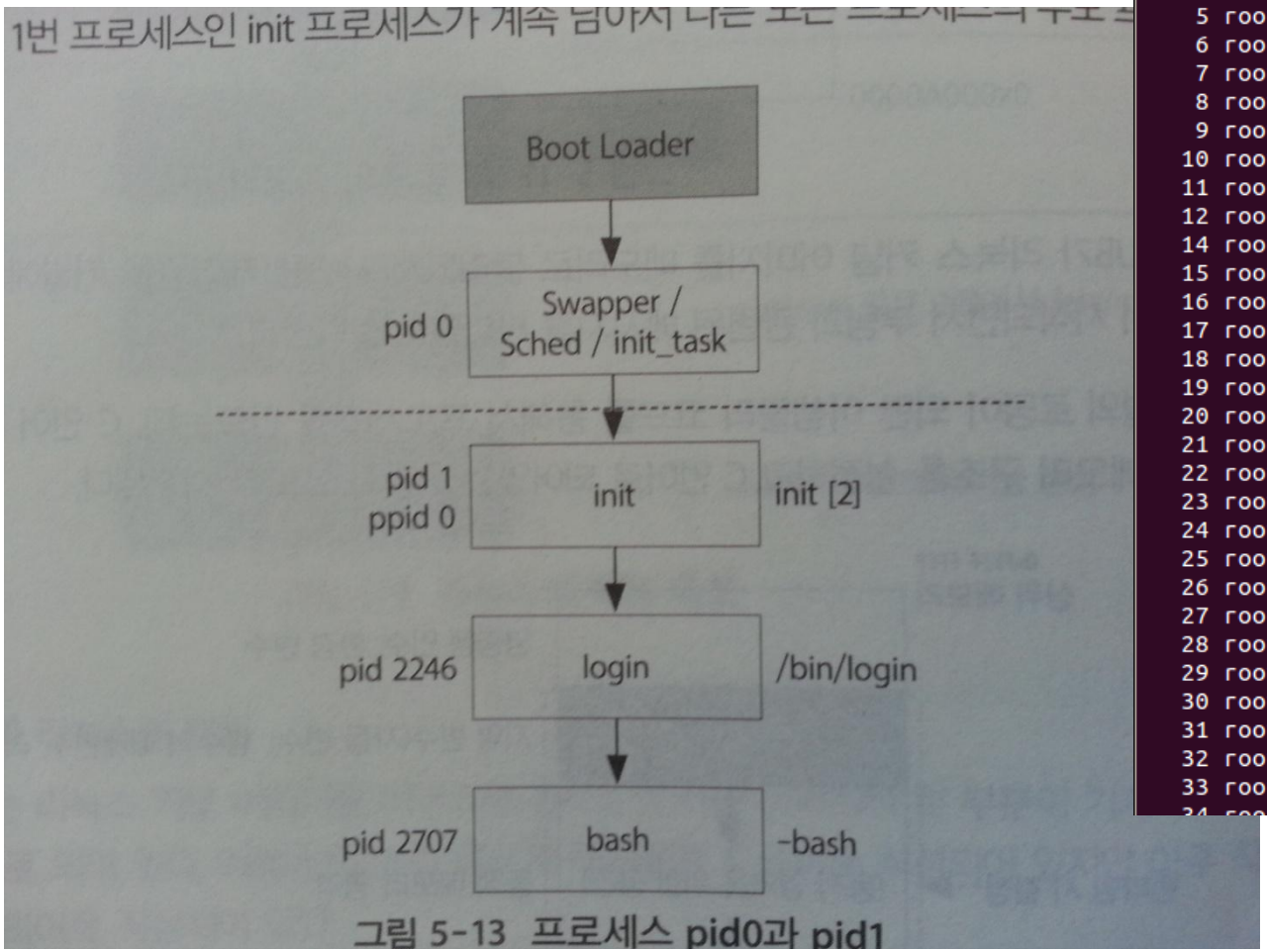
Chapter 6.

: init process & 사용자 영역 초기화 과정

1. Init process(1)



1. Init process(2)

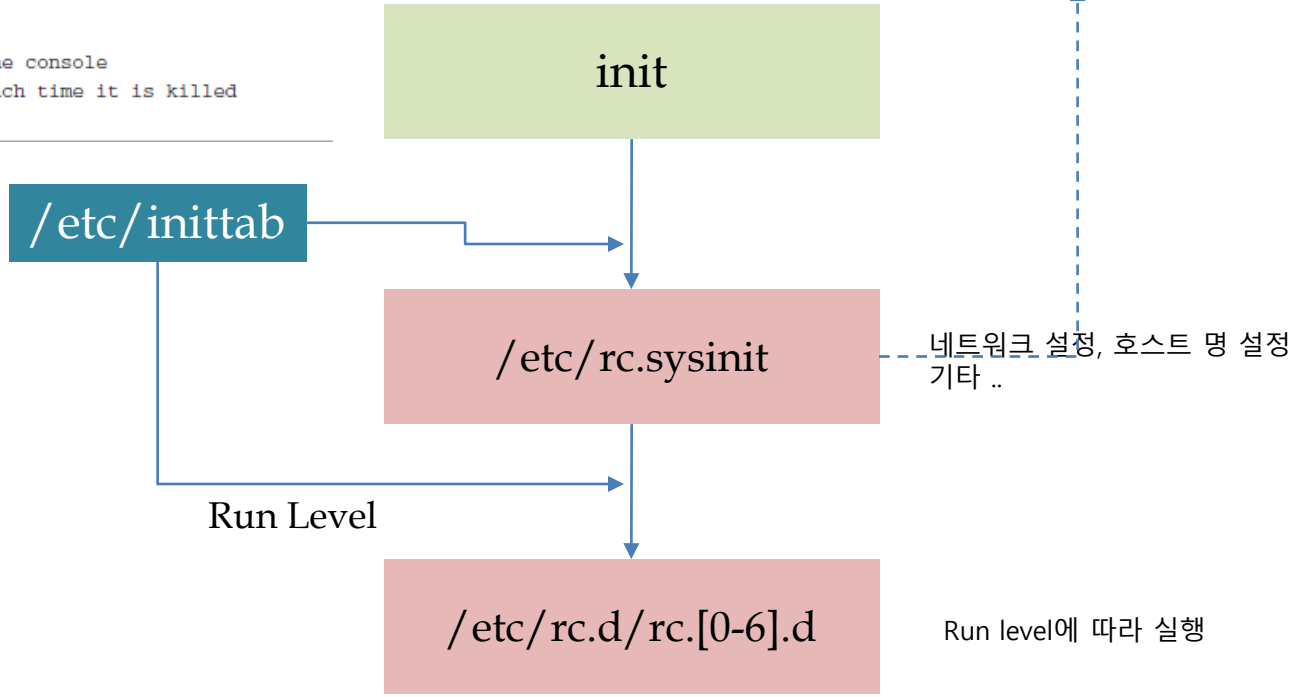



```
# ps
PID   USER     COMMAND
    1 root     init
    2 root     [kthreadd]
    3 root     [ksoftirqd/0]
    4 root     [kworker/0:0]
    5 root     [kworker/0:0H]
    6 root     [kworker/u8:0]
    7 root     [rcu_preempt]
    8 root     [rcu_sched]
    9 root     [rcu_bh]
   10 root     [migration/0]
   11 root     [migration/1]
   12 root     [ksoftirqd/1]
   14 root     [kworker/1:0H]
   15 root     [migration/2]
   16 root     [ksoftirqd/2]
   17 root     [kworker/2:0]
   18 root     [kworker/2:0H]
   19 root     [migration/3]
   20 root     [ksoftirqd/3]
   21 root     [kworker/3:0]
   22 root     [kworker/3:0H]
   23 root     [khelper]
   24 root     [kdevtmpfs]
   25 root     [netns]
   26 root     [perf]
   27 root     [khungtaskd]
   28 root     [writeback]
   29 root     [crypto]
   30 root     [bioset]
   31 root     [kblockd]
   32 root     [kworker/1:1]
   33 root     [rpciod]
   34 root     [kswapd0]
```

1. Init process(3) – inittab & runlevel(1)

```
# This is the first process (actually a script) to be run.
si::sysinit:/etc/rc.sysinit

# Execute our shutdown script on entry to runlevel 0
l0:0:wait:/etc/init.d/sys.shutdown

# Execute our normal startup script on entering runlevel 2
l2:2:wait:/etc/init.d/runlvl2.startup

# This line executes a reboot script (runlevel 6)
l6:6:wait:/etc/init.d/sys.reboot

# This entry spawns a login shell on the console
# Respawn means it will be restarted each time it is killed
con:2:respawn:/bin/sh
```

```
#!/bin/sh

echo "This is rc.sysinit"

busybox mount -t proc none /proc

# Load the system loggers
/sbin/syslogd
/sbin/klogd

# Enable legacy PTY support for telnetd
busybox mkdir /dev/pts
busybox mknod /dev/ptmx c 5 2
busybox mount -t devpts devpts /dev/pts
```



1. Init process(3) – inittab & runlevel(2)

TABLE 6-2 Runlevels

Runlevel	Purpose
0	System shutdown (halt)
1	Single-user system configuration for maintenance
2	User-defined
3	General-purpose multiuser configuration
4	User-defined
5	Multiuser with graphical user interface on startup
6	System restart (reboot)

ex) init 6

2. Root file system

TABLE 6-1 Top-Level Directories

Directory	Contents
bin	Binary executables, usable by all users on the system[1]
dev	Device nodes (see Chapter 8, "Device Driver Basics")
etc	Local system configuration files
home	User account files
lib	System libraries, such as the standard C library and many others
sbin	Binary executables usually reserved for superuser accounts on the system
tmp	Temporary files
usr	A secondary file system hierarchy for application programs, usually read-
var	Contains variable files, such as system logs and temporary configuration

The very top of the Linux file system hierarchy is referenced by the slash charac itself. For example, to list the contents of the root directory, you would type t

```
$ ls /
```

This produces a listing similar to the following:

```
root@coyote:/# ls /
bin  dev  etc  home  lib  mnt  opt  proc  root  sbin  tmp  usr  var
root@coyote:/#
```

<rootfs를 구성하는 최소 파일>

LISTING 6-1 Contents of a Minimal Root File System

```
.
|-- bin
|   |-- busybox
|   `-- sh -> busybox
|-- dev
|   `-- console
|-- etc
|   `-- init.d
|       `-- rcS
`-- lib
    |-- ld-2.3.2.so
    |-- ld-linux.so.2 -> ld-2.3.2.so
    |-- libc-2.3.2.so
    `-- libc.so.6 -> libc-2.3.2.so

5 directories, 8 files
```

3. Rootfs in Memory(1) - initrd(ramdisk)

LISTING 6-12 Contents of a Sample `initrd`

```
.
|-- bin
|   |-- busybox
|   |-- echo -> busybox
|   |-- mount -> busybox
|   `-- sh -> busybox
|-- dev
|   |-- console
|   |-- ram0
|   `-- ttyS0
|-- etc
|-- linuxrc
`-- proc

4 directories, 8 files
```

LISTING 6-11 Sample `linuxrc` File

```
#!/bin/sh

echo 'Greetings: this is 'linuxrc' from Initial Ramdisk'
echo 'Mounting /proc filesystem'
mount -t proc /proc /proc

busybox sh
```

3. Rootfs in Memory(1) - initrd(ramdisk)

```
console=ttyS0,115200 root=/dev/nfs                          \
   nfsroot=192.168.1.9:/home/chris/sandbox/omap-target      \
   initrd=0x10800000,0x14af47
```

<initrd 사용 cmdline>

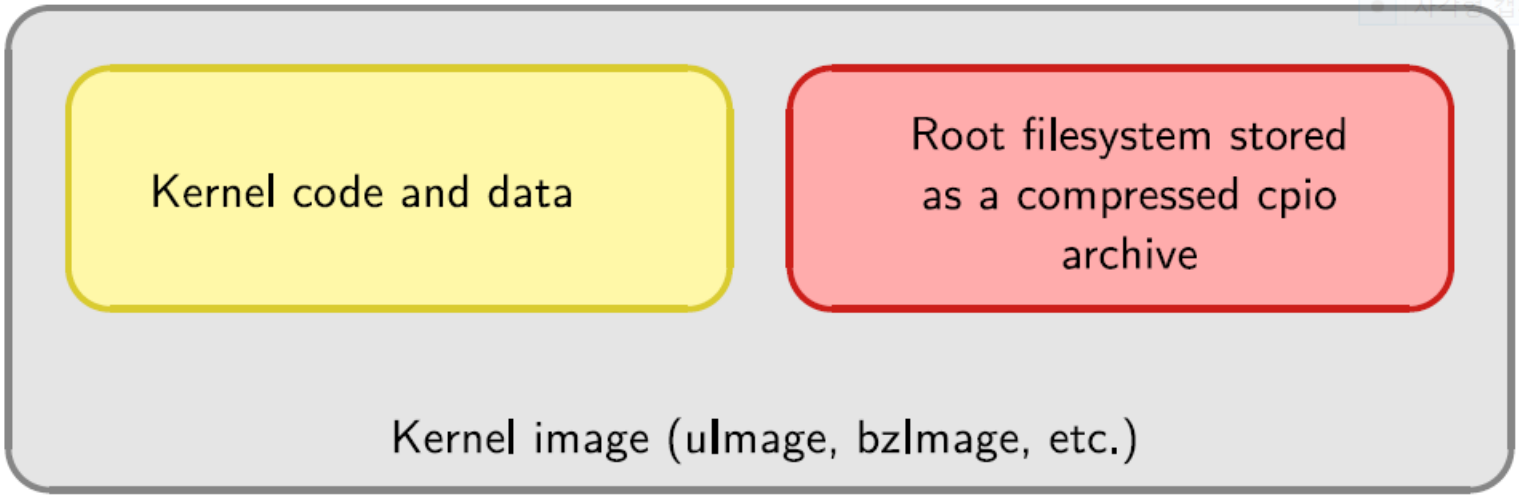
Initrd를 root file system으로 하여, 부팅 !

Initrd는 예전 방법이며,
Initramfs가 최신 방법임.

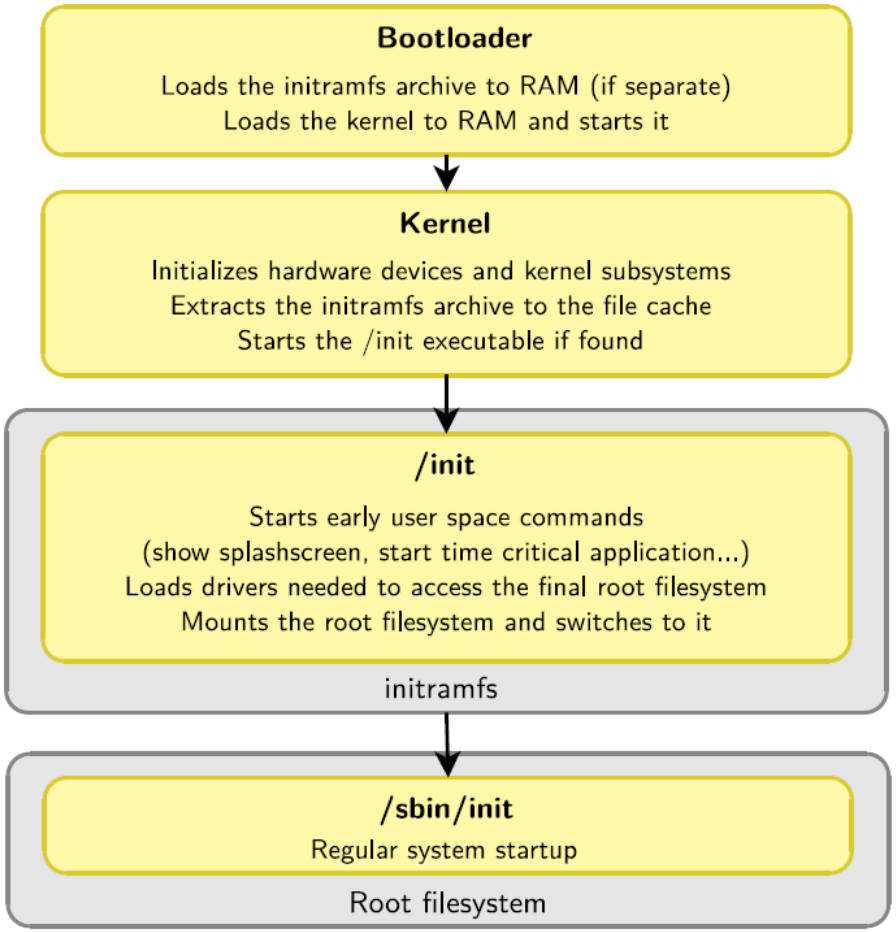
LISTING 6-10 Booting the Kernel with Ramdisk Support

```
[uboot]> tftp 0x10000000 kernel-uImage
...
Load address: 0x10000000
Loading: ############################ done
Bytes transferred = 1069092 (105024 hex)

[uboot]> tftp 0x10800000 initrd-uboot
...
Load address: 0x10800000
Loading: ########################################## done
Bytes transferred = 282575 (44fcf hex)

[uboot]> bootm 0x10000000 0x10800040
Uncompressing kernel.................done.
...
RAMDISK driver initialized: 16 RAM disks of 16384K size 1024 blocksize
...
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Greetings: this is linuxrc from Initial RAMDisk
Mounting /proc filesystem

BusyBox v1.00 (2005.03.14-16:37+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

# (<<<< Busybox command prompt)
```

3. Rootfs in Memory(2) – initramfs(1)



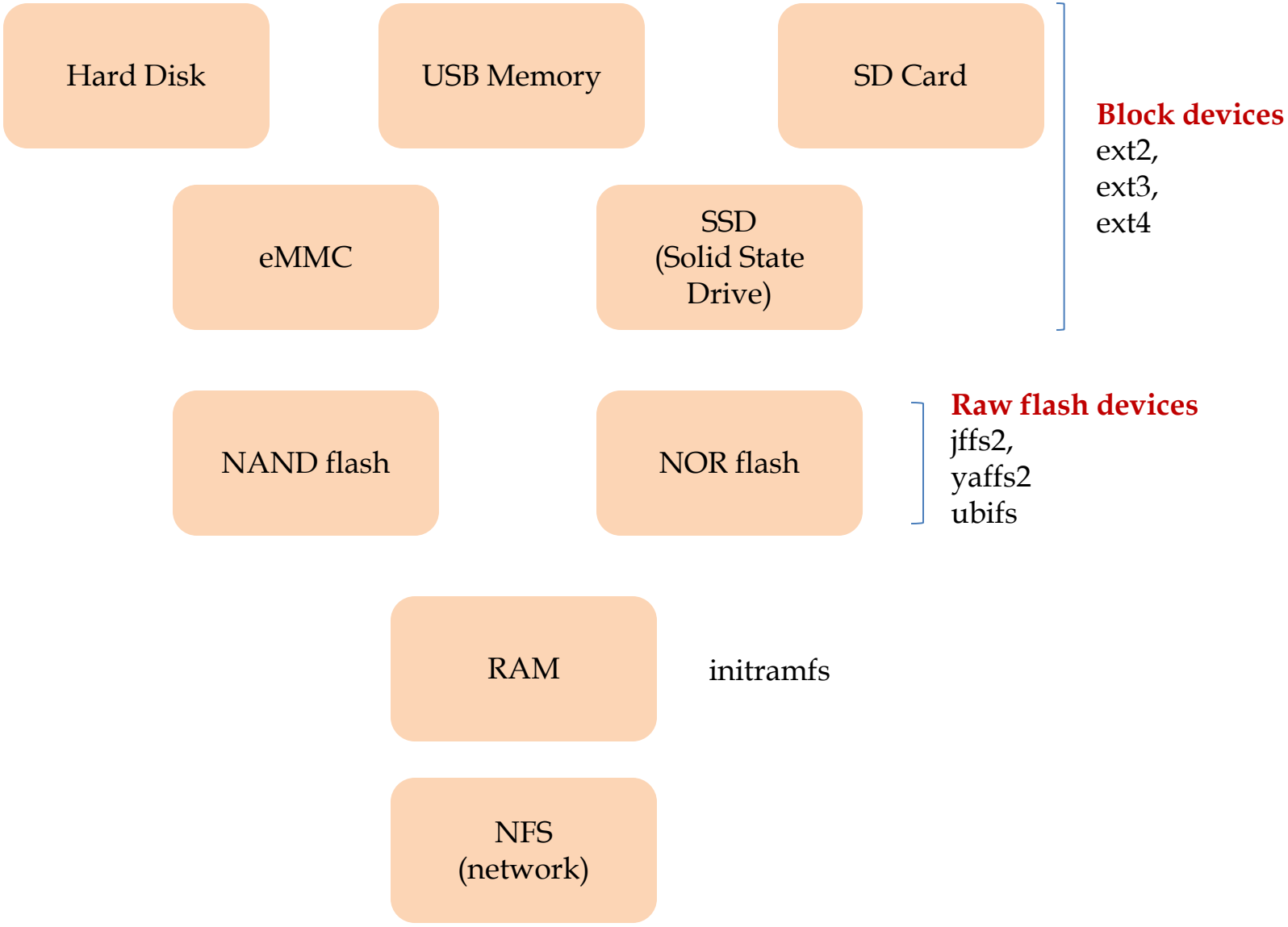
3. Rootfs in Memory(2) – initramfs(2)



Chapter 9-11.

: 주요 File systems & Busybox

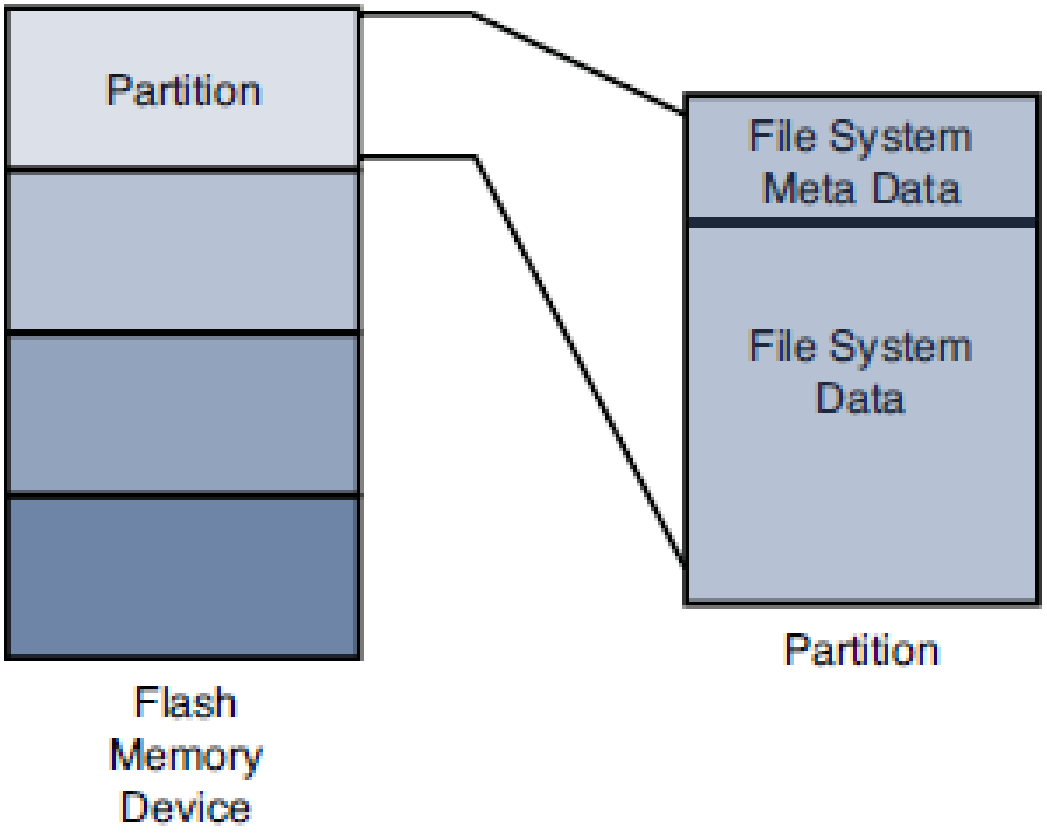
1. Block vs Flash Devices(1)



1. Block vs Flash Devices(2)

- Storage devices are classified in two main types: **block devices** and **flash devices**
  - They are handled by different subsystems and different filesystems
- **Block devices** can be read and written to on a per-block basis, without erasing.
  - Hard disks, floppy disks, RAM disks
  - USB keys, Compact Flash, SD card: these are based on flash storage, but have an integrated controller that emulates a block device, managing and erasing flash sectors in a transparent way.
- **Raw flash devices** are driven by a controller on the SoC. They can be read, but writing requires erasing, and often occurs on a larger size than the "block" size.
  - NOR flash, NAND flash

2. Partitions



3. Block File System(1) - ext2 file system 파티션

(1) 파티션 생성

LISTING 9-1 Displaying Partition Information Using ***fdisk***

```
# fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 49 MB, 49349120 bytes
4 heads, 32 sectors/track, 753 cylinders
Units = cylinders of 128 * 512 = 65536 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *           1         180       11504   83  Linux
/dev/sdb2             181         360       11520   83  Linux
/dev/sdb3             361         540       11520   83  Linux
/dev/sdb4             541         753       13632   83  Linux
```

3. Block File System(2) - ext2 file system 생성 및 mount

(2) file system 생성

LISTING 9-2 Formatting a Partition Using *mkfs.ext2*

```
# mkfs.ext2 /dev/sdb1 -L CFlash_Boot_Vol
mke2fs 1.40.8 (13-Mar-2008)
Filesystem label=CFlash_Boot_Vol
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
2880 inodes, 11504 blocks
575 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=11796480
2 block groups
8192 blocks per group, 8192 fragments per group
1440 inodes per group
Superblock backups stored on blocks:
     8193

Writing inode tables: done
Writing superblocks and filesystem accounting i

This filesystem will be automatically checked e
days, whichever comes first.  Use tune2fs -c or
```

(3) file system mount 후, 사용

```
# mount /dev/sdb1 /mnt/flash
```

(4) file 시스템 오류 검사
=> unmount 상태에서 해야 함.

LISTING 9-5 Corrupted File System Check

```
# e2fsck -y /dev/sdb1
e2fsck 1.40.8 (13-Mar-2008)
/dev/sdb1 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Inode 13, i_blocks is 16, should be 8.  Fix? yes

Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/sdb1: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdb1: 25/2880 files (4.0% non-contiguous), 488/11504 blocks
#
```

(*) ext3, ext4는 ext2에서 확장된 버전임.

4. RPi3 용 microSD 파티션 설정하기

- http://slowbootkernelhacks.blogspot.com/2016/12/booting-beaglebone-black-with-yocto.html 3절 앞 부분 참조

그림 3.1 microSD(우측) 카드와 card readder(USB type)

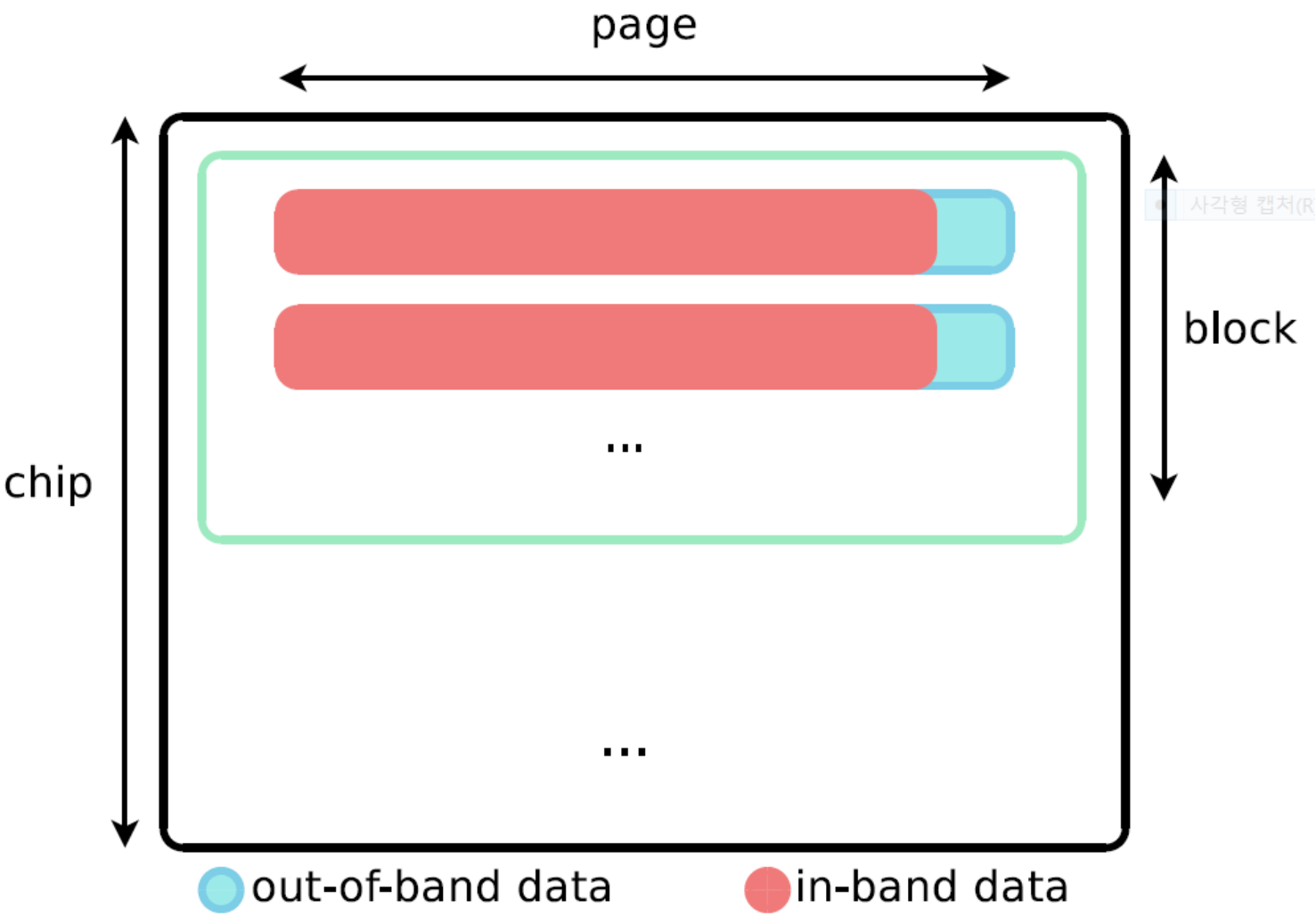
$ sudo fdisk /dev/sdb ***<= fdisk를 시작, 사전에 dmesg로 SD card 장치를 확인해 두자.***

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

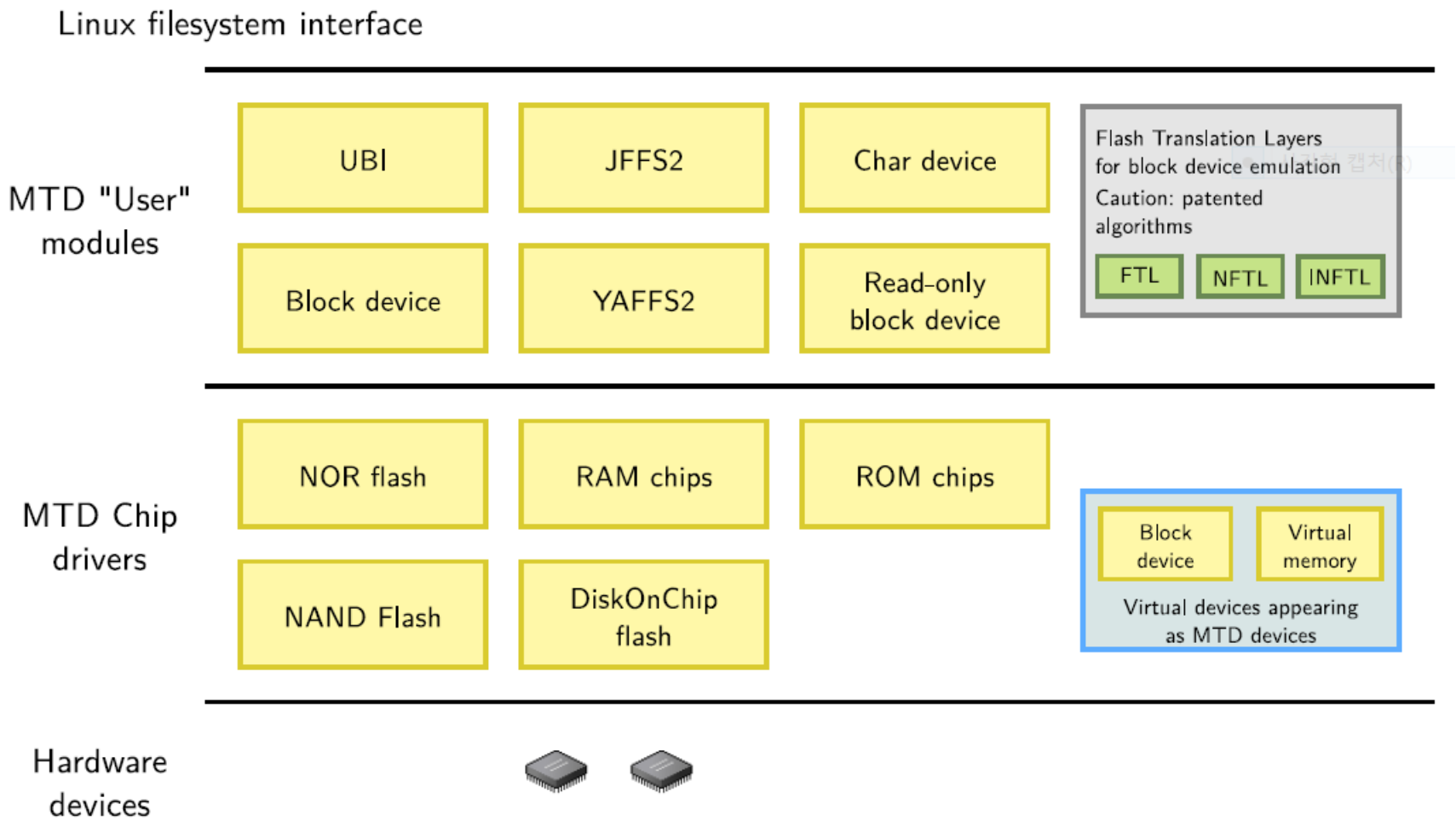
Command (m for help): **p** ***<= 현재 파티션 정보를 출력하자.***
Disk /dev/sdb: 3.7 GiB, 3965190144 bytes, 7744512 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x61d9b02b

Device Boot Start End Sectors Size Id Type

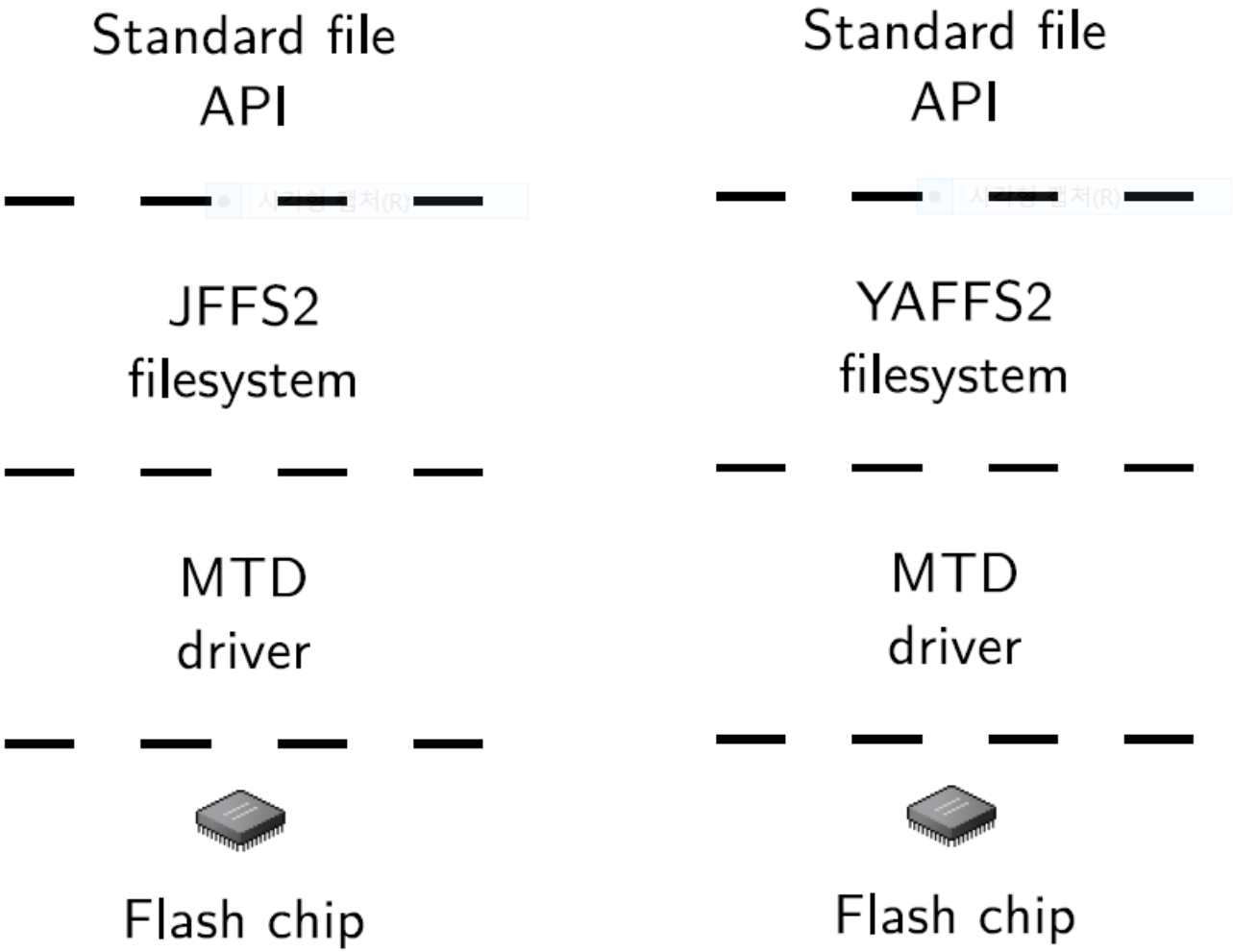
5. Flash File System – NAND flash storage



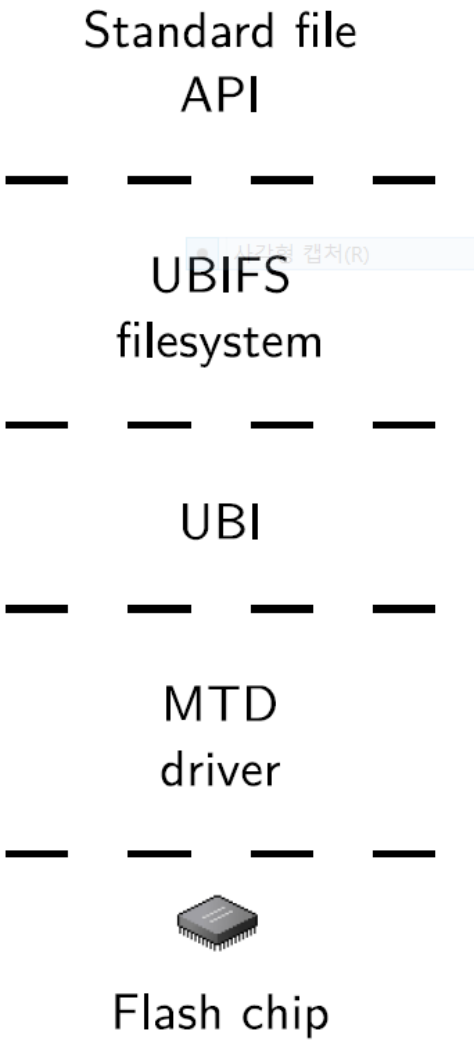
5. MTD(Memory Technology Devices)(1)



5. MTD(Memory Technology Devices)(2) - jffs2, yaffs2



5. MTD(Memory Technology Devices)(3) - UBIFS



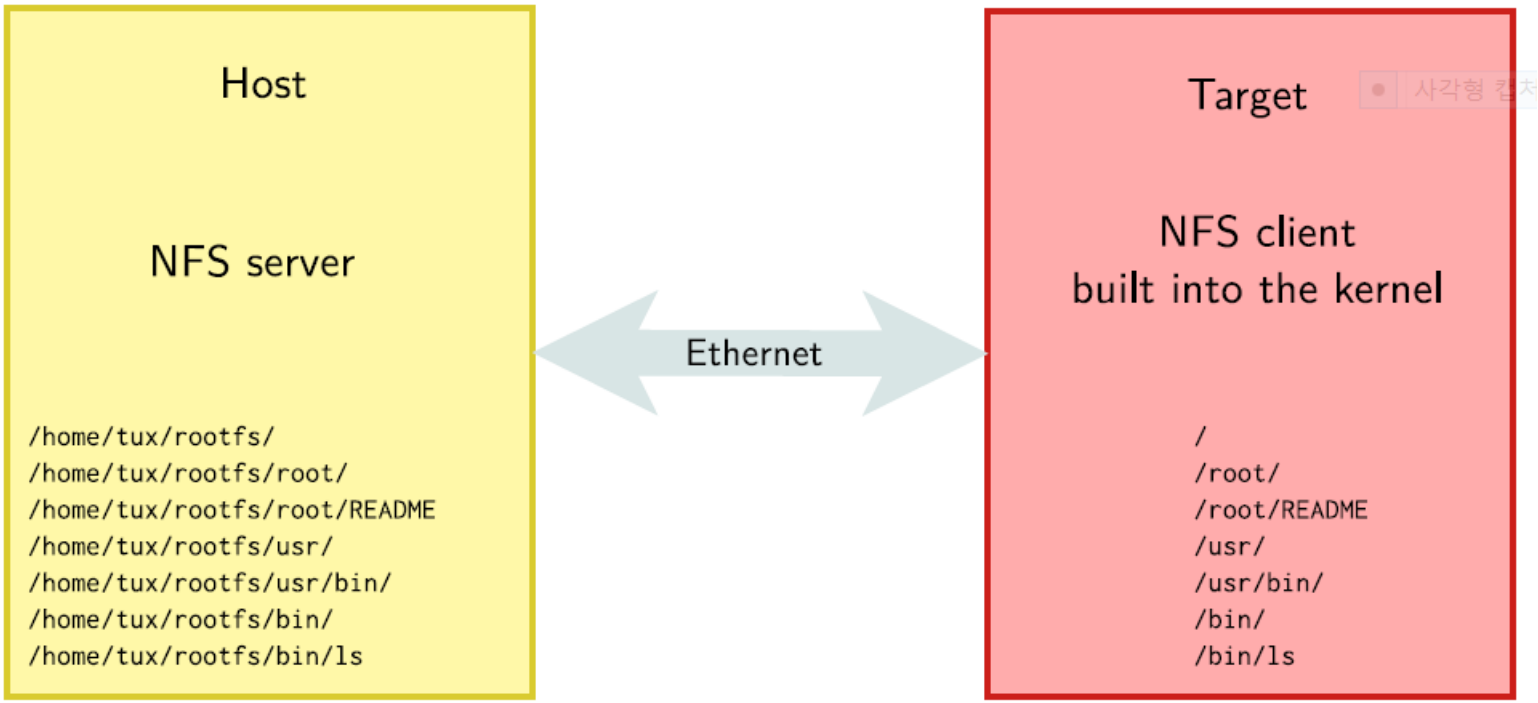
(*) 현재 flash file system 중에서 가장 안정적임 !

6. Pseudo File System – RAM file system

- /proc
- /sysfs
- /tmpfs

(*) 실제 RPi에 login하여 내용을 보여주자.

7. NFS(Network File System)



숙제 5: nfs server 설정을 해 볼 것.

8. Busybox(1)

LISTING 11-3 BusyBox Usage

```
root@coyote # busybox
BusyBox v1.13.2 (2010-02-24 16:04:14 EST) multi-call binary
Copyright (C) 1998-2008 Erik Andersen, Rob Landley, Denys Vlasenko and
others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
   or: function [arguments]...

    BusyBox is a multi-call binary that combines many common Unix
    utilities into a single executable.  Most people will create a link
    to busybox for each function they wish to use and BusyBox will act
    like whatever it was invoked as!

Currently defined functions:
    [, [[, addgroup, adduser, ar, ash, awk, basename, blkid, bunzip2,
    bzcat, cat, chattr, chgrp, chmod, chown, chpasswd, chroot, chvt,
    clear, cmp, cp, cpio, cryptpw, cut, date, dc, dd, deallocvt,
    delgroup, deluser, df, dhcprelay, diff, dirname, dmesg, du,
    dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset,
    fbsplash, fdisk, fgrep, find, free, freeramdisk, fsck, fsck.minix,
    fuser, getopt, getty, grep, gunzip, gzip, halt, head, hexdump,
    hostname, httpd, hwclock, id, ifconfig, ifdown, ifup, init, insmod,
    ip, kill, killall, klogd, last, less, linuxrc, ln, loadfont,
    loadkmap, logger, login, logname, logread, losetup, ls, lsmod,
    makedevs, md5sum, mdev, microcom, mkdir, mkfifo, mkfs.minix, mknod,
    mkswap, mktemp, modprobe, more, mount, mv, nc, netstat, nice,
    nohup, nslookup, od, openvt, passwd, patch, pidof, ping, ping6,
    pivot_root, poweroff, printf, ps, pwd, rdate, rdev, readahead,
    readlink, readprofile, realpath, reboot, renice, reset, rm, rmdir,
    rmmod, route, rtcwake, run-parts, sed, seq, setconsole, setfont,
    sh, showkey, sleep, sort, start-stop-daemon, strings, stty, su,
    sulogin, swapoff, swapon, switch_root, sync, sysctl, syslogd, tail,
    tar, tee, telnet, telnetd, test, tftp, time, top, touch, tr,
    traceroute, true, tty, udhcpc, udhcpd, umount, uname, uniq, unzip,
    uptime, usleep, vi, vlock, watch, wc, wget, which, who, whoami,
    xargs, yes, zcat
```

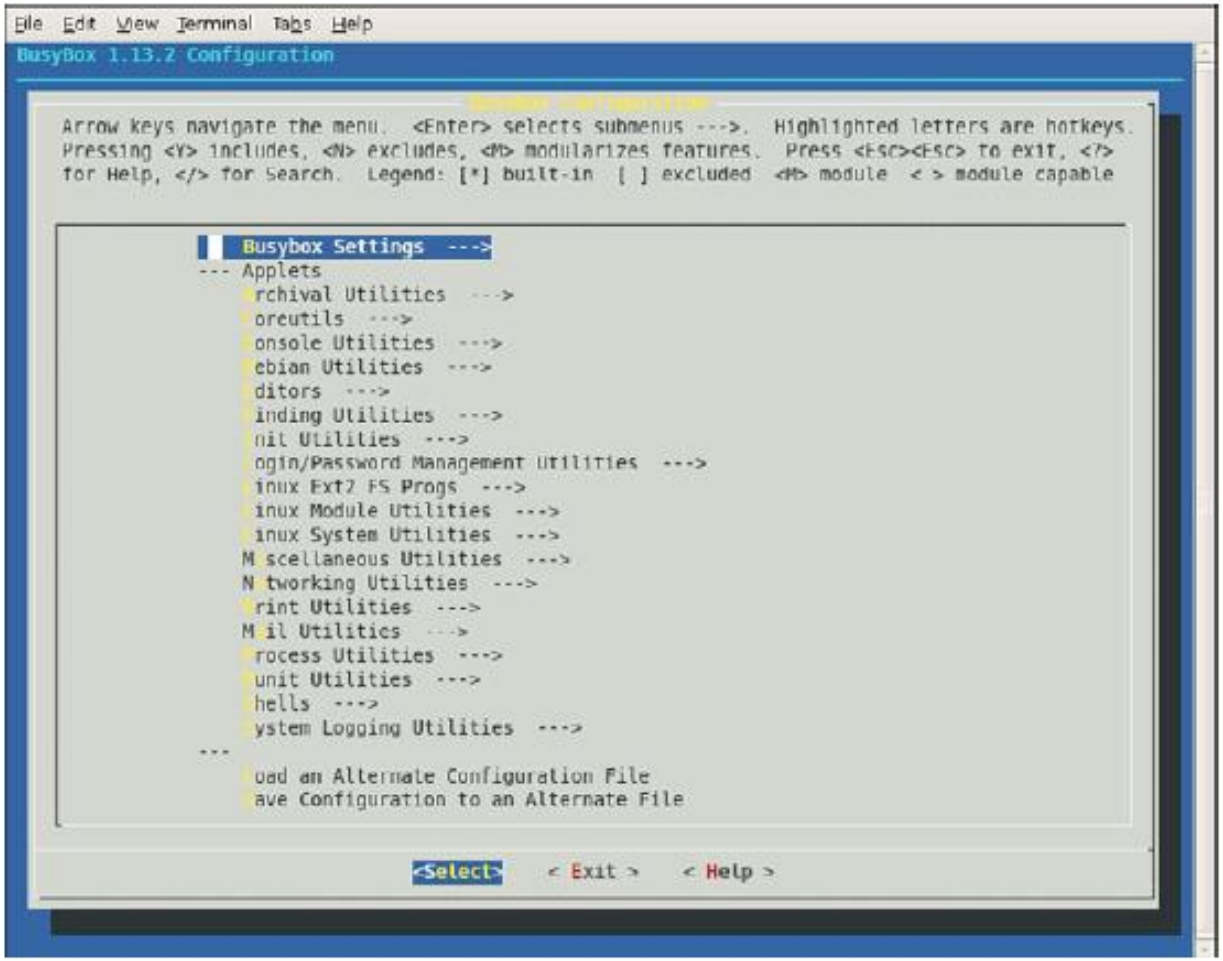
LISTING 11-5 BusyBox Symlink Structure: Tree Detail

```
[root@coyote]$ tree
.
|-- bin
|   |-- addgroup -> busybox
|   |-- busybox
```


LISTING 11-5 Continued

```
|   |-- cat -> busybox
|   |-- cp -> busybox
<...>
|   `-- zcat -> busybox
|-- linuxrc -> bin/busybox
|-- sbin
|   |-- halt -> ../bin/busybox
|   |-- ifconfig -> ../bin/busybox
|   |-- init -> ../bin/busybox
|   |-- klogd -> ../bin/busybox
<...>
|   `-- syslogd -> ../bin/busybox
`-- usr
    |-- bin
    |   |-- [ -> ../../bin/busybox
    |   |-- basename -> ../../bin/busybox
<...>
    |   |-- xargs -> ../../bin/busybox
    |   `-- yes -> ../../bin/busybox
    `-- sbin
        `-- chroot -> ../../bin/busybox
```

8. Busybox(2) - menuconfig



8. Busybox(3) – build 방법

- wget http://busybox.net/downloads/busybox-1.23.2.tar.bz2 tar -xjf busybox-1.23.2.tar.bz2

- cd busybox-1.23.2/
- export PATH=$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- defconfig
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

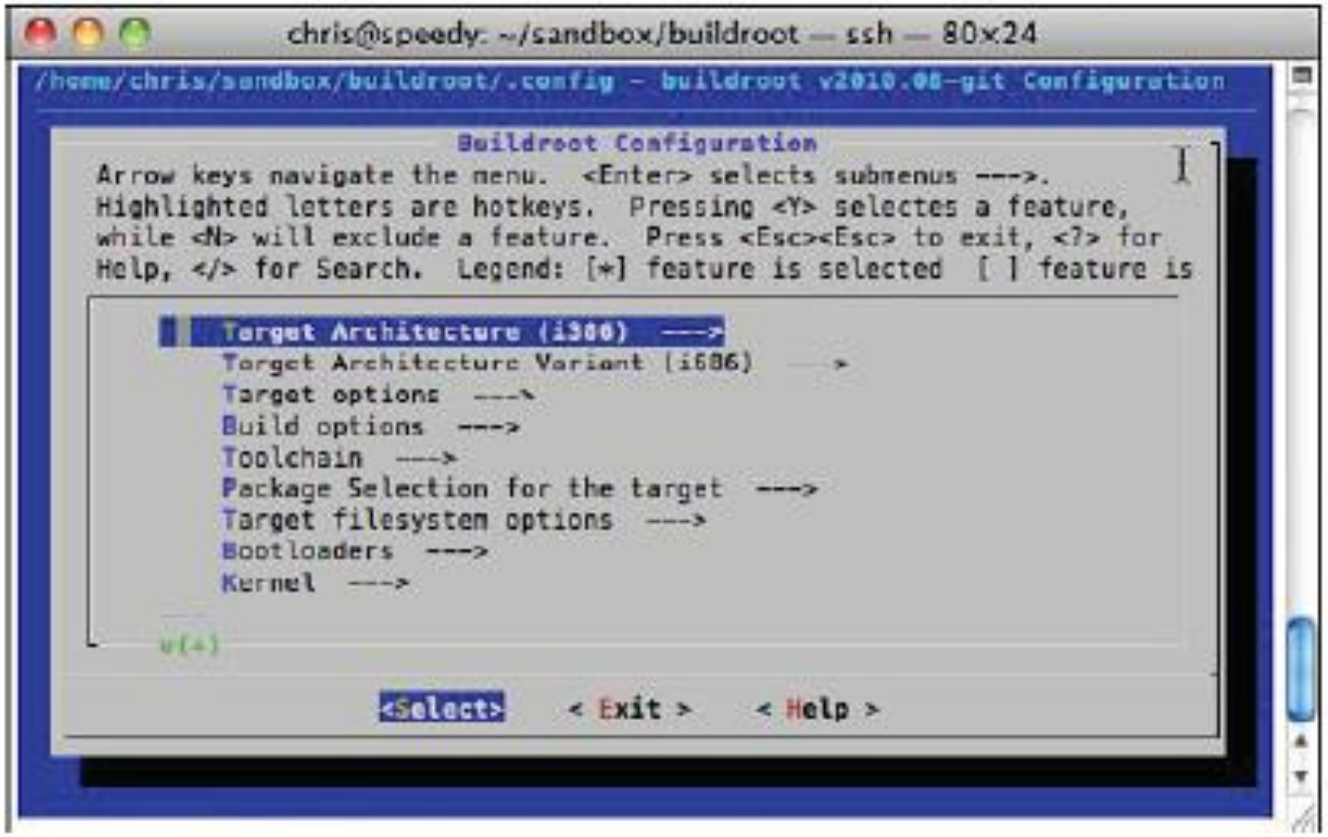
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- install CONFIG_PREFIX=$YOUR_PATH

숙제 6: busybox source를 download하고, build해 볼 것.

Chapter 16.

: build systems

1. Buildroot(1)



참고 Site: http://slowbootkernelhacks.blogspot.com/2016/12/buildroot.html

1. Buildroot(2) – RPi3

$ git clone -b jumpnow https://github.com/jumpnow/buildroot
$ cd buildroot
$ make jumpnow_rpi3_defconfig
$ make menuconfig
$ make

- 참고 Site: http://www.jumpnowtek.com/rpi/Raspberry-Pi-Systems-with-Buildroot.html

```
chyi@jupiter:~/2IP/workspace/RPi3/buildroot/output/images$ ls -l
합계 2212796
-rw-r--r-- 1 chyi chyi      23162  7월   2 12:52 bcm2709-rpi-2-b.dtb
-rw-r--r-- 1 chyi chyi      24621  7월   2 12:52 bcm2710-rpi-3-b-plus.dtb
-rw-r--r-- 1 chyi chyi      24358  7월   2 12:52 bcm2710-rpi-3-b.dtb
-rw-r--r-- 1 chyi chyi      23070  7월   2 12:52 bcm2710-rpi-cm3.dtb
-rw-r--r-- 1 chyi chyi   33554432  7월   2 12:52 boot.vfat
-rw-r--r-- 1 chyi chyi         81  7월   2 12:52 cmdline.txt
-rwxr-xr-x 1 chyi chyi      36242  7월   2 12:52 config.txt
-rw-r--r-- 1 chyi chyi    4926456  7월   2 12:52 kernel7.img
drwxr-xr-x 2 chyi chyi       4096  7월   2 12:52 overlays
-rw-r--r-- 1 chyi chyi 2013265920  7월   2 12:52 rootfs.ext2
lrwxrwxrwx 1 chyi chyi         11  7월   2 12:52 rootfs.ext4 -> rootfs.ext2
drwxr-xr-x 2 chyi chyi       4096  7월   2 12:35 rpi-firmware
-rw-r--r-- 1 chyi chyi 2046820864  7월   2 12:52 rpi3-sdcard.img
chyi@jupiter:~/2IP/workspace/RPi3/buildroot/output/images$
```

$ sudo dd if=output/images/rpi3-sdcard.img of=/dev/sdb bs=1M

- 주의: of=/dev/sdb 는 자신의 PC 환경에 따라 다를 수 있음.

1. Buildroot(3) - Booting

```
Raspbian GNU/Linux 9 michaelpi ttyAMA0
michaelpi login: [    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.14.52-v7 (chyi@jupiter) (gcc version 8.1.0 (Buildroot 2018.0
8-git-ga2e4aae)) #1 SMP Mon Jul 2 12:45:29 KST 2018
[    0.000000] CPU: ARMv7 Processor [410fd034] revision 4 (ARMv7), cr=10c5383d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[    0.000000] OF: fdt: Machine model: Raspberry Pi 3 Model B Rev 1.2
[    0.000000] Memory policy: Data cache writealloc
[    0.000000] cma: Reserved 8 MiB at 0x3ac00000
[    0.000000] percpu: Embedded 17 pages/cpu @ba34a000 s38604 r8192 d22836 u69632
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 240555
[    0.000000] Kernel command line: 8250.nr_uarts=1 bcm2708_fb.fbwidth=656 bcm2708_fb.fbhei
ght=416 bcm2708_fb.fbswap=1 vc_mem.mem_base=0x3ec00000 vc_mem.mem_size=0x40000000  root=/de
v/mmcblk0p2 rootwait rootfstype=ext4 console=tty1 console=ttyAMA0,115200
[    0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
[    0.000000] Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
[    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Memory: 940232K/970752K available (7168K kernel code, 576K rwdata, 2084K rod
ata, 1024K init, 704K bss, 22328K reserved, 8192K cma-reserved)
[    0.000000] Virtual kernel memory layout:
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
[    0.000000]     vmalloc : 0xbb800000 - 0xff800000   (1088 MB)
[    0.000000]     lowmem  : 0x80000000 - 0xbb400000   ( 948 MB)
[    0.000000]     modules : 0x7f000000 - 0x80000000   (  16 MB)
[    0.000000]       .text : 0x80008000 - 0x80800000   (8160 kB)
[    0.000000]       .init : 0x80b00000 - 0x80c00000   (1024 kB)
[    0.000000]       .data : 0x80c00000 - 0x80c90174   ( 577 kB)
[    0.000000]        .bss : 0x80c98078 - 0x80d48424   ( 705 kB)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[    0.000000] ftrace: allocating 25295 entries in 75 pages
[    0.000000] Hierarchical RCU implementation.
[    0.000000] NR_IRQS: 16, nr_irqs: 16, preallocated irqs: 16
[    0.000000] arch_timer: cp15 timer(s) running at 19.20MHz (phys).
[    0.000000] clocksource: arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0x46d987e4
7, max_idle_ns: 440795202767 ns
[    0.000007] sched_clock: 56 bits at 19MHz, resolution 52ns, wraps every 4398046511078ns
[    0.000021] Switching to timer-based delay loop, resolution 52ns
[    0.000270] Console: colour dummy device 80x30
[    0.000790] console [tty1] enabled
```

Login: root, password: jumpnowtek

1. Buildroot(4) – 참고 사항

- Output directory
  - ✓ output/images

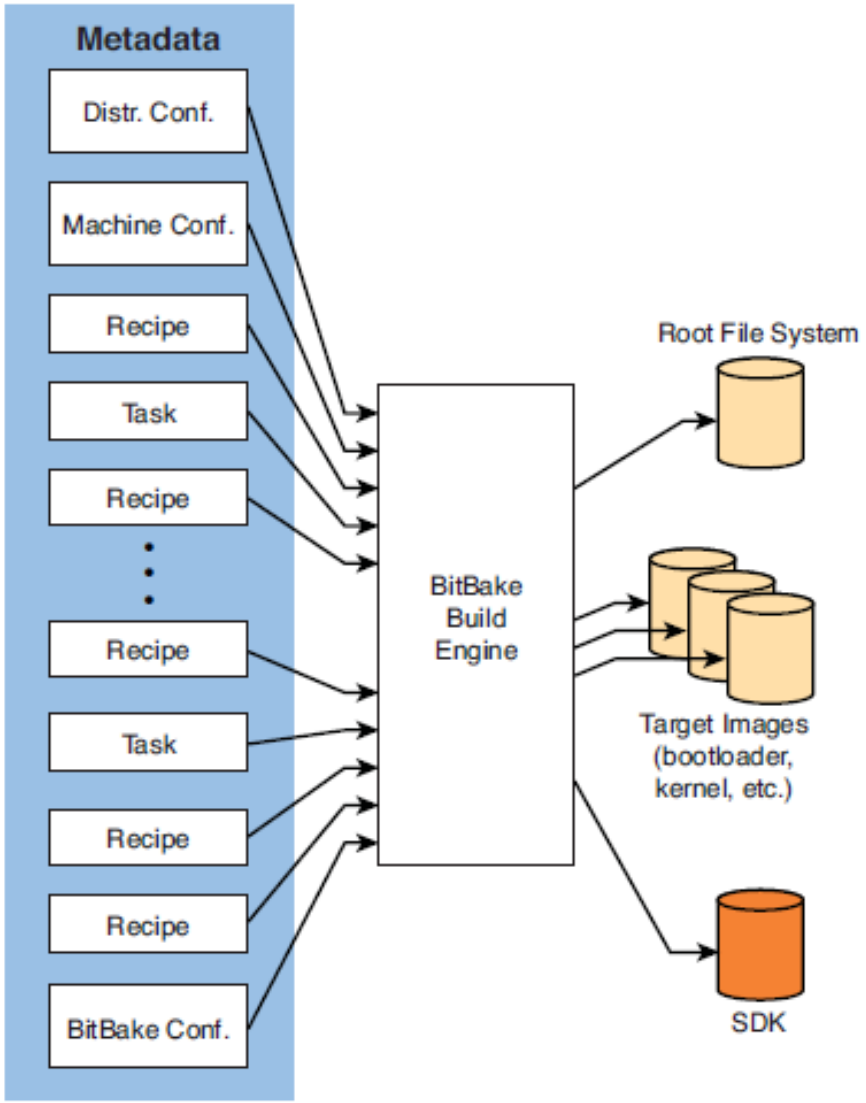
```
drwxr-xr-x 5 chyi chyi       4096  7월   3 14:01 .
drwxrwxr-x 6 chyi chyi       4096  7월   2 11:43 ..
-rw-rw-r-- 1 chyi chyi         41  7월   3 14:01 README.michael
-rw-r--r-- 1 chyi chyi      23162  7월   2 12:52 bcm2709-rpi-2-b.dtb
-rw-r--r-- 1 chyi chyi      24621  7월   2 12:52 bcm2710-rpi-3-b-plus.dtb
-rw-r--r-- 1 chyi chyi      24358  7월   2 12:52 bcm2710-rpi-3-b.dtb
-rw-r--r-- 1 chyi chyi      23070  7월   2 12:52 bcm2710-rpi-cm3.dtb
-rw-r--r-- 1 chyi chyi   33554432  7월   2 12:52 boot.vfat
-rw-r--r-- 1 chyi chyi         81  7월   2 12:52 cmdline.txt
-rwxr-xr-x 1 chyi chyi      36242  7월   2 12:52 config.txt
-rw-r--r-- 1 chyi chyi    4926456  7월   2 12:52 kernel7.img
drwxr-xr-x 2 chyi chyi       4096  7월   2 12:52 overlays
-rw-r--r-- 1 chyi chyi 2013265920  7월   3 14:00 rootfs.ext2
lrwxrwxrwx 1 chyi chyi         11  7월   2 12:52 rootfs.ext4 -> rootfs.ext2
drwxr-xr-x 2 chyi chyi       4096  7월   2 12:35 rpi-firmware
-rw-r--r-- 1 chyi chyi 2046820864  7월   2 12:52 rpi3-sdcard.img
```

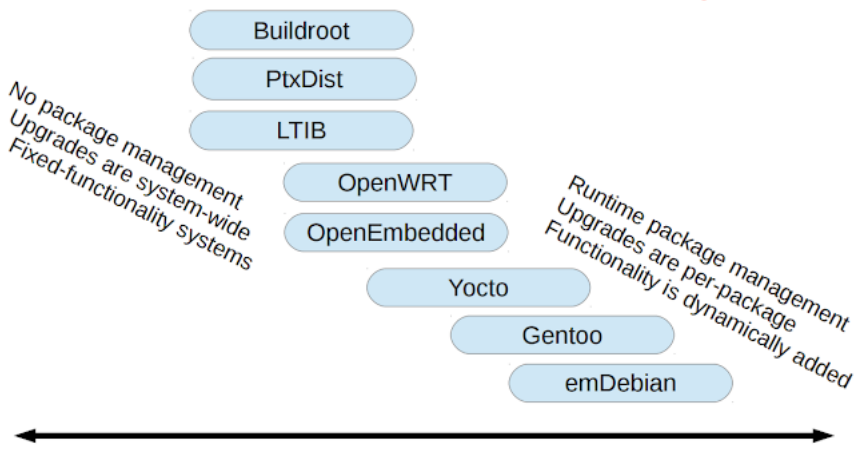
- Linux kernel source
  - ✓ output/build/linux-11dc…….

- Toolchain path
  - ✓ output/host/bin

- …

2. Yocto Project(1)





http://slowbootkernelhacks.blogspot.com/2016/12/yocto-project.html

2. Yocto Project(2) – Recipe Example

LISTING 16-4 Simple OpenEmbedded Recipe: ***hello_1.0.0.bb***

```
DESCRIPTION = "Hello demo project"
PR = "r0"
LICENSE = "GPL"

SRC_URI = "http://localhost/sources/hello-1.0.0.tar.gz"

SRC_URI[md5sum] = "90a8ffd73e4b467b6d4852fb95e493b9"
SRC_URI[sha256sum] = "fd626b829cf1df265abfceac37c2b5629f2ba8fbc3897add29f-
9661caa40fe12"

do_install() {
        install -m 0755 -d ${D}${bindir}
        install -m 0755 ${S}/hello ${D}${bindir}/hello
}
```

2. Yocto Project(3) – BitBake

LISTING 16-5 BitBake Hello Recipe Processing

```
chris@speedy:~/sandbox/build01$ bitbake hello
<...>
NOTE: Executing runqueue
NOTE: Running task 10 of 38 (ID: 5, NOTE: Running task 10 of 38 (ID: 5,
/hello_1.0.0.bb, do_fetch)
NOTE: Running task 11 of 38 (ID: 0, /hello_1.0.0.bb, do_unpack)
NOTE: Running task 15 of 38 (ID: 1, /hello_1.0.0.bb, do_patch)
NOTE: Running task 16 of 38 (ID: 7, /hello_1.0.0.bb, do_configure)
NOTE: Running task 17 of 38 (ID: 8, /hello_1.0.0.bb, do_compile)
NOTE: Running task 18 of 38 (ID: 2, /hello_1.0.0.bb, do_install)
NOTE: Running task 19 of 38 (ID: 10, /hello_1.0.0.bb, do_package)
NOTE: Running task 25 of 38 (ID: 13, /hello_1.0.0.bb, do_package_write_ipk)
NOTE: Running task 26 of 38 (ID: 9, /hello_1.0.0.bb, do_package_write)
NOTE: Running task 29 of 38 (ID: 3, /hello_1.0.0.bb, do_populate_sysroot)
NOTE: Running task 30 of 38 (ID: 12, /hello_1.0.0.bb, do_package_stage)
NOTE: Running task 37 of 38 (ID: 11, /hello_1.0.0.bb, do_package_stage_all)
NOTE: Running task 38 of 38 (ID: 4, /hello_1.0.0.bb, do_build)
NOTE: Tasks Summary: Attempted 38 tasks of which 25 didn't need to be rerun and 0
failed.
```

3. OpenWRT and Ubuntu

- http://wiki.espressobin.net/tiki-index.php?page=Software+HowTo

- <OpenWRT Project>
- https://openwrt.org/

- <ARMBIAN Project>
- https://www.armbian.com/

숙제

숙제

- 1) Serial console 설정해보기
- 2) RPi3용 ARM toolchain 설치하기
- 3) 예제 C code를 만들어서 cross-compile해 보기
- 4) RPi3용 kernel source download 받아 build해 보기
- 5) kernel module programming 해 보기(example 돌려 보기)
- 6) RPi3용 buildroot download 받아 build해 보기
- 7) 6단계에서 build한 img 파일을 microSD card에 설치하고 부팅해 보기

References

[1] Embedded Linux Primer 2nd Edition, Hallinan, Prentice Hall

[2] 사물인터넷 리눅스 프로그래밍 with 라즈베리파이, 서영진, Jpub

[3] embedded-linux-slides.pdf, free electrons
=> *http://free-electrons.com/doc/training/embedded-linux/*

[4] petazzoni-device-tree-dummies.pdf, free electrons
=> *https://events.linuxfoundation.org/sites/events/files/slides/petazzoni-device-tree-dummies.pdf*

[5] 코드로 알아보는 ARM 리눅스 커널, Jpub

[6] http://free-electrons.com/doc/training/linux-kernel/linux-kernel-slides.pdf

추천 도서

<Linux kernel & device drivers programming>

[1] 리눅스 디바이스 드라이버, 한빛 미디어(저자: 유영창)
=> 오래된 책이나, 유용함.

[2] Linux Device Drivers 3rd Edition, Oreilly
=> 한글 번역판 있음.

[3] Linux Kernel Development 3rd Edition - Addison Wesley 출판사(저자: Rovert Love)
=> 한글 번역판 있음. 이와 유사한 주제를 다루는 기타 교제도 가능함.

[4] 아트멜 스튜디오와 아두이노로 배우는 ATmega328 프로그래밍 - Jpub 출판사(저자: 허경용)
=> ATmega328이 목적이 아니라, Embedded의 기초를 확립하기 위한 내용이 잘 기술되어 있어, 소개함.
=> CPU 및 주변 장치(UART, SPI, I2C, GPIO, Interrupt 등) 개념 이해

<Linux 사용자 영역 programming>

[1] 유닉스 리눅스 프로그래밍 필수 유틸리티 - 한빛미디어 출판사(저자: 백창우)
=> Linux 개발 환경(vim, gcc, ld, make, subversion 등) 숙지 차원 !
=> 단, 여기에 git, bug tracking system, Yocto 등 추가 소개 필요함

[2] Advanced programming in the UNIX environment - Addison Wesley 출판사(저자: Rechard Stevens & Rago)

[3] Linux System Programming - Oreilly 출판사(저자: Rovert Love)
=> Userspace 영역과 관련된 내용으로, 이와 유사한 주제를 다루는 기타 교제도 가능함.