2012011658 김진권

부트로더

부트 스트랩(Bootstrap) 코드라고도 불리며, 커널 실행에 필요한 환경을 설정하고, os 이미지를 메모리에 복사하는 일을 한다.

x86 pc의 경우 boot loader는 MBR(Master boot record)에 존재하며, 1섹터로 이루어져 있다. 또한 마지막 2 바이트가 0x55,0xAA로 시그니처가 존재한다. 따라서, bios에서 마지막 2바이트가 0x55,0xAA인 디바이스들을 찾은 후 해당 boot loader을 메모리에 올리고, boot loader는 커널 실행에 필요한 환경을 설정하고, 커널 이미지를 메모리에 복사한다.

u-boot (universal bootloader) 는 임베디드 디바이스에서 많이 쓰이며, 68k, ARM, AVR32, Blackfin, MicroBlaze, MIPS, Nios, SuperH, PPC,x86과 같은 다양한 아키텍처를 지원한다.

리눅스 피시의 경우 lilo(linux loader)라고 불리는 부트 로더가 존재한다. x86 피시와 마찬가지로 MBR에 존재하며 0x55,0xAA 시그니처를 가진다. 현재는 grub gnu 부트로더로 대부분 운영 체제의 커널을 불러올 수 있으며, 인자를 넘겨 줄 수 도 있다.

크로스 컴파일

크로스 개발 환경은 4가지로 구분이 가능하다.

1. native build

build host target 환경이 모두 동일한 경우

2. cross build

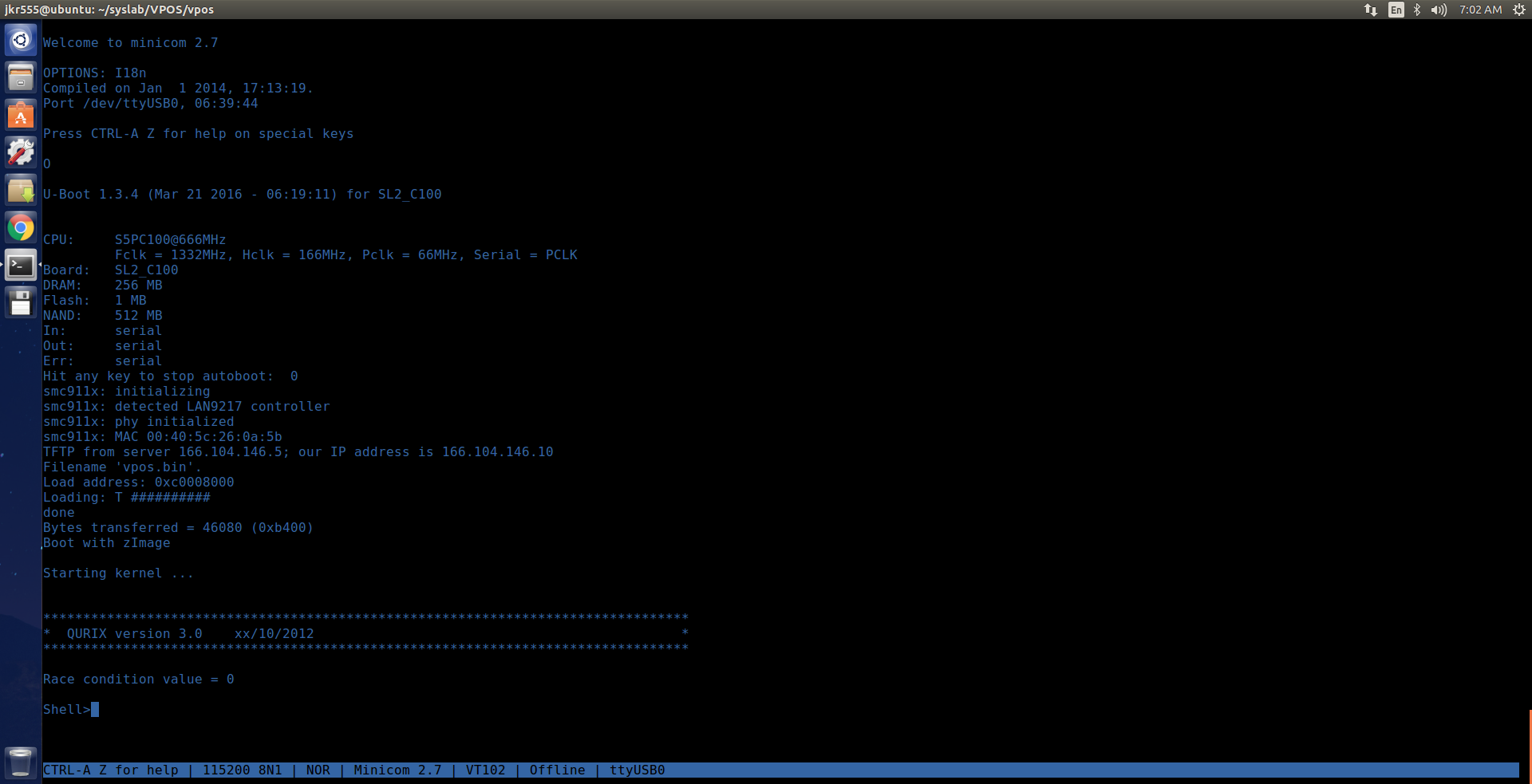
build host 가 동일하고 target이 다른 경우, 주로 임베디드 환경에서 사용되며, 코드를 pc에서 작성하고 target용 컴파일을 pc에서 하며, 동작은 target에서 하는 경우

3. cross-native build

build 가 다르며, host target이 동일한 경우, compile 시간이 오래걸려서 compile을 외부서버에서 하는 경우

4. Canadian build

build host target이 모두 다 다른 경우이다.



#include "../include/vh\_cpu\_hal.h"

.extern VPOS\_kernel\_main

.extern vk\_undef\_handler

.extern vh\_hwi\_classifier

.extern vk\_swi\_classifier

.extern vk\_pabort\_handler

.extern vk\_dabort\_handler

.extern vk\_fiq\_handler

.extern vk\_not\_used\_handler

.extern vk\_irq\_test

.extern vk\_sched\_save\_tcb\_ptr

.global vh\_VPOS\_STARTUP

.global vk\_save\_swi\_mode\_stack\_ptr

.global vk\_save\_swi\_current\_tcb\_bottom

.global vk\_save\_irq\_mode\_stack\_ptr

.global vk\_save\_irq\_current\_tcb\_bottom

.global vk\_save\_pabort\_current\_tcb\_bottom

.global vh\_restore\_thread\_ctx

.global vh\_save\_thread\_ctx

.global vh\_save\_ctx\_bottom

.equ vh\_USERMODE, 0x10

.equ vh\_FIQMODE, 0x11

.equ vh\_IRQMODE, 0x12

.equ vh\_SVCMODE, 0x13

.equ vh\_ABORTMODE, 0x17

.equ vh\_UNDEFMODE, 0x1b

.equ vh\_MODEMASK, 0x1f

.equ vh\_NOINT, 0xc0

.equ vh\_INT, 0x00

.equ vh\_userstack, 0x21200000

.equ vh\_svcstack, 0X21400000

.equ vh\_irqstack, 0x21600000

.equ vh\_abortstack, 0x21800000

.equ vh\_undefstack, 0x21a00000

.equ vh\_fiqstack, 0x21c00000

.equ vh\_vector\_base, 0x20008044// relocated vector table base address

.equ vh\_VICBASE, 0xe4000000

.text

vh\_VPOS\_STARTUP:

/\* Camouflaged code for imitating linux

Linux has a header that includes 8 nop operation, branch code, magic number, binaray file start offset, and file size(end offset \*/

nop

nop

nop

nop

nop

nop

nop

nop

b vh\_VPOS\_reset

magicn: .long 0x016f2818 // Linux magic number

startn: .long 0x00000000 // start address(offset) is 0

endn: .long 0x0000d8fc // end address(offset) is file size(byte)

/\* Camouflaged code end \*/

nop

nop

nop

nop

vh\_vector\_start:

b vh\_VPOS\_reset

b vk\_undef

b vh\_software\_interrupt

b vh\_pabort

b vk\_dabort

b vk\_not\_used\_handler

b vh\_irq

b vk\_fiq\_handler

vh\_VPOS\_reset:

// variable initialization

mov r0,#0x00

str r0,vk\_save\_swi\_mode\_stack\_ptr

str r0,vk\_save\_swi\_current\_tcb\_bottom

str r0,vk\_save\_irq\_mode\_stack\_ptr

// Mask interrupt and fast interrupt

mrs r0, cpsr

orr r0,r0,#0xc0

msr cpsr,r0

// Invalidate all instruction caches to point of unification. Also flushes branch target cache.

mov r0,#0x00

mcr p15,0,r0,c7,c5,0

// Control Register Setting

mrc p15,0,r0,c1,c0,0

bic r0,r0,#0x01

bic r0,r0,#0x04

bic r0,r0,#0x1000

bic r0,r0,#0x2000

orr r0,r0,#0x02

orr r0,r0,#0x800

mcr p15,0,r0,c1,c0,0

// change vector table base address (0x20008044)

ldr r0, =vh\_vector\_base

mcr p15, 0, r0, c12, c0, 0

// stack pointer setting

mrs r0,cpsr

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_UNDEFMODE|vh\_NOINT

msr cpsr\_cxsf,r1

ldr sp,=vh\_undefstack

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_ABORTMODE|vh\_NOINT

msr cpsr\_cxsf,r1

ldr sp,=vh\_abortstack

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_IRQMODE|vh\_NOINT

msr cpsr\_cxsf,r1

ldr sp,=vh\_irqstack

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_FIQMODE|vh\_NOINT

msr cpsr\_cxsf,r1

ldr sp,=vh\_fiqstack

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_SVCMODE|vh\_NOINT

msr cpsr\_cxsf,r1

ldr sp,=vh\_svcstack

// user mode sp

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

orr r1,r0,#vh\_USERMODE|vh\_INT

msr cpsr\_cxsf,r1

ldr sp,=vh\_userstack

b VPOS\_kernel\_main

코드 설명

bic r0,r0,#vh\_MODEMASK|vh\_NOINT

-> vh\_NOINT|vh\_MODEMASK 결과는 1101 1111이 된다.

bic는 r0에 r0&~(1101 1111) 값을 넣어 주며, 1101 1111에서1에 값에 해당하는 r0 값을 0으로 clear 해준다.

orr r1,r0,#vh\_USERMODE|vh\_INT

-> vh\_USERMODE|vh\_INT 결과는 0001 0000이 되며, r0와 이 값을 or 시킨 후 r1에 대입시킨다.

msr cpsr\_cxsf,r1

-> r1의 값을 cpsr에 대입한다.

ldr sp,=vh\_userstack

-> sp 값을 vh\_userstack 으로 load 한다.