임베디드 시스템 설계

실습 과제 8회차

2015004493

김형순

1. Bootloader

Bootloader 는 일반적으로 컴퓨터나 임베디드 시스템이 처음 시작 할 때 저장되어 있는 OS 를 load 하여 메모리에 mount 해주어 OS 를 가동 시키는 프로그램이다. 또한 bootloader 는 OS 시작 전에 메모리를 초기화, 하드웨어 초기화 등의 역할을 수행하기도 한다.

컴퓨터나 임베디드 시스템의 전원을 킬 경우 컴퓨터는 BIOS 실행 시킨다. BIOS 는 컴퓨터의 모든 입력과 출력을 관리하는 펌웨어로 BIOS 가 실행이 되면 입출력에 필요한 기능들이 활성화 돼 있는지 확인, 초기화한다(POST). POST 가 끝난 후 어떤 운영체제로 컴퓨터를 시작 할 결정하기 위해 bootloader 가 포함되어 있는 MBR 섹터를 읽어온 후 어떤 bootloader 를 선택 할 지 결정하게 된다. 그 후 결정된 운영체제의 bootloader 는 실제 운영체제가 시작되기 전 필요한 stack, heap 등의 주소 공간을 설정하고 하드웨어를 초기화 한 후 OS 를 memory 에 mount 하는 역할을 수행한다.

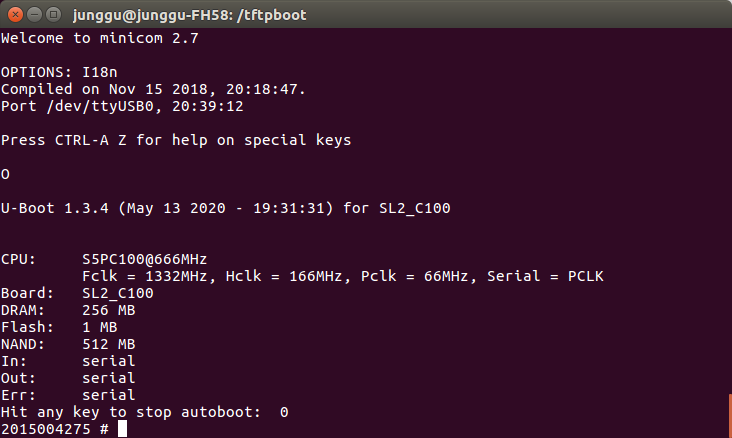
우리의 실습에서는 ARM, PowerPC 등 기반의 임베디드 보드에 적합한 U-Boot 라는 Bootloader를 사용한다. 이번 실습을 통해 u-boot 를 이용해 nand 의 c0008000 번지에 vpos커널을 포팅을 해주어 시스템을 부팅할 수 있도록 해줬다.

2. 크로스 컴파일

: 크로스 컴파일이란 target 머신에서 동작 할 수 있는 바이너리를 host 머신에서 대신 컴파일 하는 것이다. 크로스 컴파일과 반대되는 개념인 native 컴파일을 하면 컴파일 된 binary 는 컴파일을 한 환경과 비슷한 환경, 즉 같은 종류의 OS 나 CPU 등 platform 에서만 작동 되도록 compile 된다. 예를들어 x86 과 linux 를 사용하는 플랫폼 에서 컴파일한 binary 는 Mac OS 나 Widows 등 다른 OS 를 사용하거나 ARM 이나 MIPS 등 다른 CPU 를 사용하는 환경에서는 동작하지 않는다. 보통 임베디드 시스템의 경우 pc 에 비해 성능이 떨어지거나 운영체제 자체가 지원되지 않는 마이크로 컨트롤러에서는 개발과 컴파일을 하기 어렵다. 따라서 개발과 컴파일을 임베디드 시스템에서 직접 하지 않고, 성능이 좋고 편리성이 높은 다른 플랫폼에서 개발하기 위해 크로스 컴파일을 사용한다.

크로스 컴파일을 하기 위해서는 크로스 컴파일러가 필요한데, 이 크로스 컴파일러는 소프트웨어 개발에 필요한 도구들로 이루어진 toolchain 이라는 컴파일러 환경에 포함되어 있다. 가장 대표적인 toolchain 은 GNU 가 있다. GNU 는 x86, ARM v4, v5, v6, v7, MIPS 등의 binary 를 모두 생성 할 수 있는 크로스 컴파일러를 포함하고 있다. 예시로 “arm -none -linux -gnueabi” 은 리눅스에서 동작하는 GNU EABI(Embedded Application Binary Interface) 를 사용하는 ARM toolchain 이다.

3. U – boot 실행 화면 (출력된 학번은 조원(2015004275 강정구)의 학번 입니다.)



(VPOS 부팅 화면은 다음장에)

4. VPOS 부팅 화면

