uC/OS-II 뛰어넘기(I)

임베디드 OS 개관

임베디드 CPU의 고성능화와 집적화에 의해 많은 임베디드 시스템에 OS가 적용되고 있다. 그 중 WinCE와 Linux가 많은 사람들의 관심을 받으며 여러 시스템에 적용되고 있다. 하지만, 이러한 상황에서도 꿋꿋이 자신의 영역을 지키며 임베디드 시장의 일부를 차지하고 있는 RTOS에 대해서 이해하고, 그것을 가지고 충분히 활용할 수 있도록 한걸음씩 나아가 보자.

글: 김대홍/CyberLab 실장, 삼성 첨단기술연수소 RTOS 강사
redizi@armkorea.com

Embedded System에서의 OS

근래에 들어 많은 시스템에 OS를 적용하기 위한 시도들이 많이 진행되고 있다. Linux, Windows CE, RTOS 등 아마도 임베디드 시스템을 접한 독자라면 이 중의 하나정도는 들어보지 않았을까 하는 생각이 들 정도로 임베디드 시스템에서 OS는 보편화 되고 있는 추세이다.

사실, 몇 년 전까지만 해도 임베디드 시스템에 OS를 적용한다는 것은 그렇게 일반화되어 있지는 않았다. 보통 OS를 생각할 때, 그 크기나 기능을 생각해보면 임베디드 시스템에 적용하기에 CPU의 성능이나 주변 환경이 PC에 비해 상당히 능력이 떨어져서 OS를 적용하기에 상당히 무리가 있었지만, CPU가 32 bit화 되고 그 성능이 강화되고 주변장치들이 CPU로 집적이 이루어지면서 OS의 적용이 이루어지기 시작했다. 그 선두주자 중의 하나가 바로 앞에서 다뤄졌던 ARM이다.

물론, 고성능 임베디드용 CPU로 MPC 계열이나 MIPS 계열도 있지만, 핸드폰이나 PDA 같은 단말기에 많은 적용이 이뤄지면서 시장을 주도하게 된 것은 ARM이다. 이러한 시장에서부터 32 bit 프로세서가 적용이 되고, 많은 기능을 수행하기 위해서 OS가 필요해진 것은 당연한 일이 아닐까?

앞으로도 이러한 OS의 적용이 일반화가 될 것이라는 사실에는 아무도 의심치 않으리라 생각한다.

OS란 무엇인가?

그렇다면 OS가 하는 일이 무엇이기에 OS를 적용하기 위해 이렇게 애를 쓰고 있는 것일까?

임베디드 시스템은 그 특성상 같은 구조나 같은 주변장치를 갖는 경우가 드물다(물론, 같은 부분들도 많지만…). 그래서 새로운 시스템을 개발하게 되면, 밑바닥부터 프로그램밍 해가는 방식을 취하게 된다. 임베디드 시스템을 개발해 온 독자라면 이러한 개발의 어려운 점들을 여실히 느낄 것이라 생각한다. 그래도 다행히 기존의 시스템들은 이런 개발 방식을 취하더라도 충분히 관리하고 운영할 수 있었다. 하지만, 현재 적용되는 제품들은 네트워크, GUI(Graphic User Interface), USB 등 많은 주변장치들을 필요로 하게 됐으며 임베디드 시스템에서도 이러한 장치들이 기본 사양으로 바뀌어가면서 거의 PC와 같은 주변장치들을 운영해야 될 필요성이 나타나게 됐다.

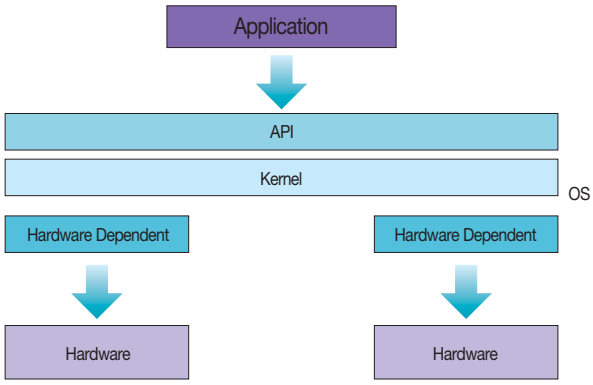
예로, 인터넷 냉장고를 생각해보자. 기존의 냉장고는 온도조절 잘되고 전력만 적게 소비하면 좋은 제품이었다. 지금은 그러한 기능들은 기본이고 인터넷 연결도 지원되며 GUI로 제어가 되면서 원격으로도 제어가 가능해야 한다. 그렇다면, 이것을 개발하는 개발자들은 TCP/IP, Network Driver, GUI 환경 등 모든 것을 처음부터 제작을 해야 할까? 이 제품의 사양이 바뀐다면? 상위 프로그램의 유지관리는?

이러한 문제들을 해결해 줄 수 있는 해결책은 바로 OS이다.

1) OS는 일관된 플랫폼을 지원한다.

OS는 서비스를 지원하기 위한 API를 지원한다.

예로 PC의 Windows 개발자는 Windows CE 응용 프로그램을 작성하기 위해 처음부터 모든 것을 배워야 할까? 그렇지 않다. MS에서 제공하는 Win32 API란 기본 함수를 사용한다면 PC의 윈도우 응용 프로그램을 짤 수 있는 프로그래머는 좀 더 쉽게 Windows CE 응용 프로그램을 작성할 수 있게 된다. 물론, 이러한 상황은 이상적인 경우로 시스템 및 OS 차이에 의해 100% 호환이란 것은 없지만 그래도 처음부터 모든 것을 공부하는 것보다는 훨씬 낳지 않을까?



하드웨어가 변경되어도 끄떡없다.(이상적으로…)

2) 상위 응용 프로그램을 쉽게 유지관리 할 수 있다.

486에서 제작했던 윈도우용 프로그램을 펜티엄에서 동작시키는 것은 아주 간단하다. 이유는 당연히 OS가 같기 때문이다. 그렇다면 X란 하드웨어에 A란 OS를 적용한 시스템으로 제작한 응용 프로그램이 있을 때, 이것을 Y란 하드웨어로 변경시 A란 OS를 적용한다면 이 응용 프로그램을 새로운 시스템에서 빠르게 적용이 가능할 것이다.

3) 주변장치들을 관리해 준다.

현재 임베디드 시스템에 적용되는 CPU에는 많은 장치들이 붙어있다. Timer, Real Time Clock, ADC, Touch Panel Controller, LCD Controller, LAN, USB, IIS, IIC, NAND Controller 등 엄청나게 많은 장치들이 붙어있다. 그렇다면 이 많은 장치들을 어떻게 효율적으로 관리할 것인가? LAN으로 파일을 전송하면서 USB Keyboard를 통해 입력을 받고, 그 결과를 TFT-LCD로 출력하는 프로그램을 짠다고 하면 벌써부터 머리가 아파 올 것이다.

새로운 시스템을 개발하는데 있어서의 가장 많이 고려하는 문제 중의 하나가 주변장치이다. 이것은 여러 개의 주변장치들을 동시에 관리하고 안정적으로 운영하는 데에 있어서 어려움을 겪기 때문이다. 하지만, OS가 있기에 이러한 문제들을 손쉽게 처리할 수 있게 된다. OS는 이러한 장치들을 사용자의 간섭 없이 효율적으로 관리해주면서 사용자에게는 손쉽게 장치에 접근할 수 있도록 API를 제공해 주게 된다.

또한, 디바이스 드라이버라는 개념을 통해 모든 디바이스를 체계적으로 관리함으로써 개발 및 유지를 손쉽게 지원해 주게 된다. 즉, 한번 작성된 디바이스 드라이버를 같은 OS를 사용하는 다른 시스템으로 쉽게 포팅을 할 수 있게 된다.

4) 메모리를 관리해준다.

응용 프로그램이나 디바이스 드라이버를 작성하면서 메모리를 사용하는 것은 당연한 일이다. 기존의 OS가 없는 시스템의 경우 메모리를 사용할 때, 보통 전역변수로 설정하여 프로그래머가 상황에 맞게 관리를 해주어야 했다. 하지만, 응용 프로그램이 커지고 주변장치들이 사용하는 메모리들이 늘어나면 메모리의 관리 효율이 떨어지게 된다. 만약 1M의 RAM 영역을 잠시 사용하는 응용 프로그램의 경우, 이 때를 위해 메모리를 영구적으로 할당해 주어야 하는 경우가 발생할 수도 있게 된다. 하지만, 이것도 OS에게 관리를 맡기게 되면 좀더 손쉽게 효율적으로 메모리를 사용할 수 있는 응용 프로그램을 작성할 수 있게 된다.

5) 프로토콜 및 각종 드라이버들을 제공한다.

LAN을 사용하기 위해 LAN 칩에 액세스하는데 성공했다고 해보자. 아마도 일이 거의 끝났다고 생각하는 분들은 없을 것이다. 실제로 이것을 사용하기 위해서는 프로토콜 스택이 필요하기 때문이다. 그렇다면 이것을 처음부터 개발한다면 개발자 나름대로 스트레스와 개발 일정에 차이게 되고 제품의 출시가 늦어지면서 경쟁력이 떨어지게 될 것이다. 하지만, 이 문제 또한 OS에게 맡기자. 내가 제작한 것보다는 훨씬 훌륭한 프로토콜 스택을 제공해준다.

USB의 경우를 살펴보자. 요즘 기존의 장치들이 USB로 바뀌어감에 따라 임베디드 시스템에서도 USB는 좀 더 보편적인 버스가 될 것이다. 그 만큼 매력도 있지만. 문제는 디바이스 드라이버이다. 이미 USB를 다뤄본 개발자들이라면 이러한 부분들이 아직은 손쉽게 해결할 수 있는 부분이 아니라는 것을 알 것이다. 호스트쪽에서는 USB 디바이스를 관리해주고 통신하기위한 기본적인 스택이 올라가야하고 디바이스 입장에서는 그에 맞는 디바이스 드라이버를 제작해주어야 한다. 그렇지만 이러한 부분도 OS가 해결해 줄 수 있다. 물론 PCI와 같은 버스도 이러한 범주에 포함이 된다.

위에서 정리한 내용들 외에도 OS가 지원해주는 여러 가지 장점들이 많다. 그렇다고 해서 OS가 만능은 아니다. OS를 사용함에 의해 발생될 수 있는 비용의 상승과 전체적인 시스템의 크기가 커질 수 있는 단점들도 발생할 수 있다. 그러므로 이러한 장단점들을 정확하게 파악하여 필요한 OS가 무엇인지, OS의 적용이 적절한지를 판단할 수 있도록 충분한 사전 검토가 필요하다.

OS의 종류

현재 임베디드 시스템에서 많은 적용이 이루어지고 있는 OS는 대표적으로 아래 3가지로 정리할 수 있다.

1) RTOS

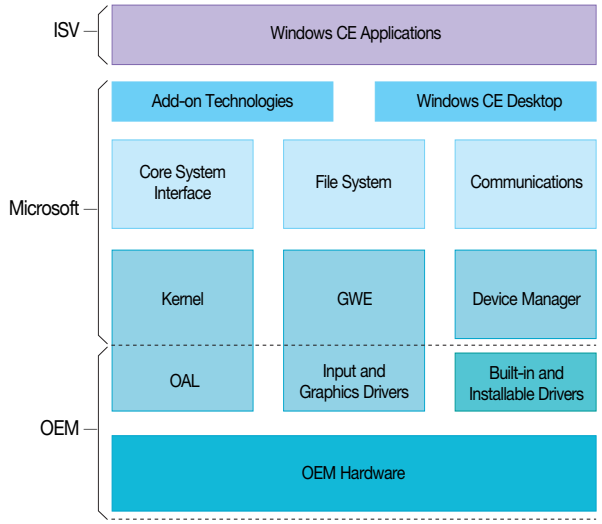
임베디드 OS의 개념이 일반화되기 전부터 오랫동안 임베디드 시장에서 독보적인 존재로서 아직도 그 존재를 무시할 수 없는 OS이다. RTOS는 제어분야와 같이 신뢰성과 정확성을 요하는 시스템에 많이 적용되어 왔다. 핵발전소, 미사일, 엔진제어와 같은 특수 분야에서 셀룰러폰까지 그 분야를 말로 다하기 어려울 정도로 모든 분야에서 사용되고 있다. 현재는 WinCE나 Linux에 의해 그 시장들이 위협을 받고 있으나 RTOS만이 가지고 있는 기능과 성능으로 맞서가고 있다.

2) Windows CE

많은 사람들이 알고 있는 Microsoft사의 임베디드 OS이다. 암호명 페가수스로 1996년 처음 발표된 이후로 현재는 Windows CE 4.2까지 발표되었다.

CE의 가장 큰 장점이라면 역시 GUI가 아닐까 싶다. 기존의 PC 사용자라면 쉽게 접근할 수 있고, 강력한 멀티미디어 환경은 상당한 이점이 있으며 PC와의 호환성을 강력하게 지원하면서 임베디드 OS 시장에서 점점 그 시장을 확보해 나가고 있다.

하지만, 고사양의 환경을 요구하기 때문에 단가 문제와 안정성에 대해 충분히 검증되지 않은 OS이기 때문에 제어분야에 적용하기에는 아직 무리가 많이 있다. 이러한 약점을 MS에서도 파악하고 Real Time 기능 등을 추가해가면서 계속 대응해 나가고 있기 때문에 앞으로의 발전성에 많은 기대를 해본다.



http://www.microsoft.com/windows/embedded/ce.net/

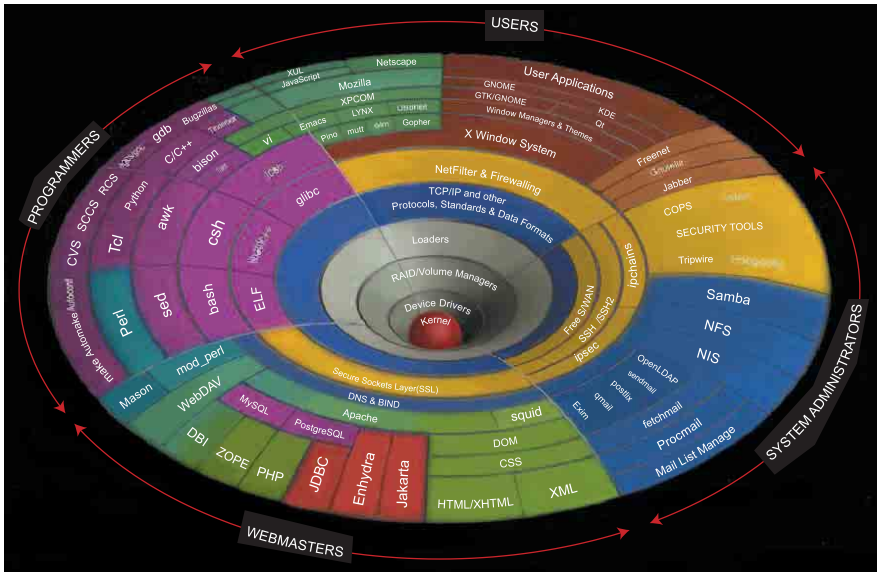
3) Linux

그 이름만 들어도 무언가를 기대하게 하는 OS이다. PC에서 이미 그 기능과 성능을 인정받고 있으며, 거기에 임베디드 시장에까지 적용이 되면서 많은 사람들의 이목을 받는 OS가 아닌가 싶다. 수많은 해커들에 의해 개발 및 유지가 되고 있는 만큼 발전도 빠르고 로열티가 없다는 장점때문에 많은 회사들이 적용을 검토하고 있는 OS이다.

하지만, 여기에 함정이 있다. linux는 공짜가 맞다. 하지만 개발은 공짜가 아니다. 앞에서 언급한 바와 같이 임베디드 시스템은 그 시스템의 구성이 천차만별이기 때문에 항상

포팅이란 부분을 염두해 두어야 하고 linux라는 커널의 안정성과 신뢰성을 누구도 책임져주지 않는다. 그렇다면 커널에 책임감 있는 회사가 있어야 하지 않을까? 직접 개발을 한다면 그러한 신뢰성 부분을 직접 책임지고 나아가야 한다. 그 누구도 불안정한 시스템을 싫어할 테니까 말이다.

Linux는 개발툴 및 환경면이나 그 내용을 볼 때, 정말 매력적인 OS이긴 하다. 하지만, 신뢰성을 요구하는 임베디드 시스템에 공짜라는 이유로 무조건적인 적용을 검토하기에는 무리수가 있다는 것은 분명하다.



리눅스 시스템의 해부

모든 OS가 마찬가지이긴 하지만, 비용을 지불한다는 것은 그만큼에 대한 책임을 지겠다라는 다른 의미이기도 하다.

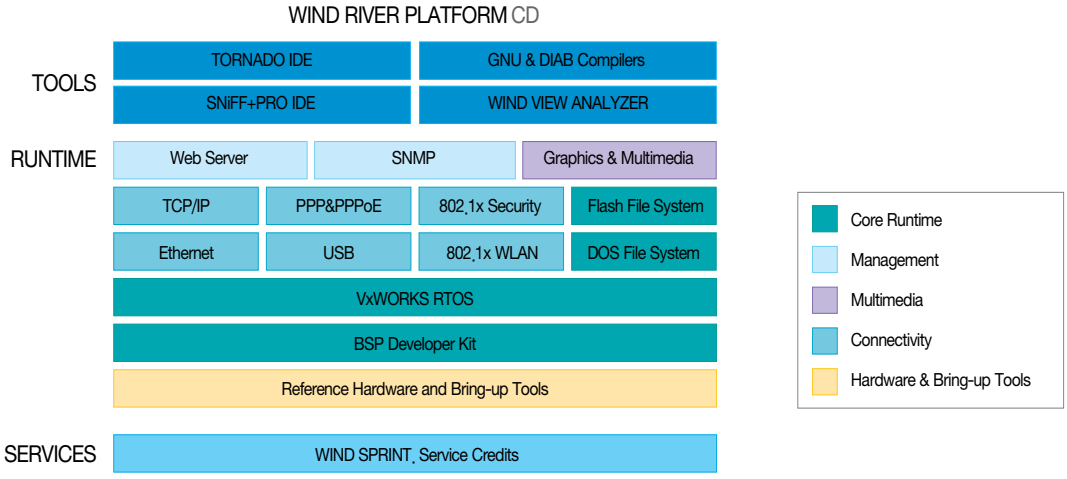
RTOS란 무엇인가?

앞에서 다룬 3가지 OS 중에서도 우린 RTOS에 대해서 좀 더 알아보기로 하겠다.

RTOS란 Real Time Operating System의 약자로 말 그대로 Real Time, 즉 실시간성을 가진 OS이다. 그렇다면 Windows CE나 Linux는 Real Time OS인가?

Windows CE는 Real Time 기능을 가지고 있다고 말을 한다. 물론 Real Time 기능을 위해 용감하게 Windows CE를 사용하는 사람은 아직까지 주위에서 보진 못했다.

Linux는 커널 자체가 Real Time을 위한 기능을 지원하



WIND RIVER Platform for Consumer Devices 블록 다이어그램

지 않는다. 하지만, 이 기능을 위한 프로젝트가 존재하고 상용으로 지원하는 회사들도 있다.

http://www.mvista.com
http://www.lynuxworks.com
http://www.fsmlabs.com
http://www.realtimelinuxfoundation.org

임베디드 시스템에서 중요시 하는 기능 중에 하나가 바로 이 Real Time성인데, 정확하게 말하면 실시간 응답성이라고 말할 수 있다.

이 의미는 일반 PC를 사용해 본 독자라면 쉽게 이해할 수 있으리라 생각이 된다. 예를 들어, 프로그램을 더블 클릭해서 실행시켰는데 한참 있다가 실행되는 경우가 있는데, 이것은 유저의 요구에 실시간적으로 응답을 못한 경우이다. 만약 유저의 요구에 언제나 실시간적으로 실행이 된다면 실시간 응답성을 가진다고 말을 할 것이다.

이와 같이 시스템이 어떤 요구에 실시간적으로 응답을 할 수 있는 기능이 Real Time이다. 이러한 기능은 임베디드 시스템에서는 상당히 중요한 요소로서 일반 PC의 OS에서는 볼 수 없는 기능이다.

그렇다면, 왜 이러한 기능이 중요할까? 예를 들어, 엘리베이터에 OS를 적용하는 경우를 생각해보자.

엘리베이터에 OS를 올려서 동작시킬 때 어떤 사고에 의해 엘리베이터의 케이블이 끊어지면서 비상 시스템이 실시간으로 동작해야 하는 상황을 생각해보자. 만약, Windows 98과 같은 OS를 올린다면 실시간적인 응답은 고사하고, 파란화면이 뜨지 않는 것을 감사해야 할 것이다. 물론 이러한 시스템에 그러한 OS를 쓰는 사람은 없을 것이다. 여기서 중요한 것은 실시간적인 응답과 신뢰성에 대해 이야기하고자 하는 것이다. 제어 분야에서 특히 이러한 실시간적인 응답과 신뢰성을 보장해 주어야 하는 경우가 많은데, 이러한 요구를 충족시키는 것이 바로 RTOS이다. 또한 RTOS는 WinCE나 Linux 계열에 비해 아주 작은 용량의 메모리만을 필요로 하고, 고성능의 하드웨어를 필요로 하지 않으므로 많은 시스템에 적용이 가능하다. 그렇기에 WinCE와 Linux도 이러한 부분들을 충족시키고자 노력을 하고 있는 것이다.

특징

- Portable
  - 다양한 시스템에 적용이 가능하다.
- ROMable
  - ROM에 전체 커널을 다 올릴 수있다.
- Preemptive
  - 선점형 커널을 가지고 있다.
- Multitasking
  - 멀티태스킹을 지원한다.
- Deterministic
  - 실행시간을 예측할 수 있다.
- Robust & Reliable
  - 신뢰성이 있다.

여기까지해서 일반적인 임베디드 OS에 관해서 설명을 해봤다. 다음 시간에는 RTOS의 하나인 uC/OS를 소개하고 실습을 통하여 RTOS에 대한 감을 익혀 보도록 하겠다. RTime

■ RTOS 제품관련 사이트

- VxWorks
  http://www.windriver.com
- pSOS
  http://www.windriver.com
- QNX
  http://www.qnx.com
- Nucleus
  http://www.acceleratedtechnology.com
- TRON
  http://www.sakamura-lab.org/TRON
- OS-9
  http://www.radisys.com/microware.cfm
- Lynx
  http://www.lynuxworks.com
- Velos
  http://www.velos.co.kr
- eCos
  http://sources.redhat.com/ecos
- uC/OS
  http://www.ucos-ii.com