사례 발표

모바일 기기를 위한 임베디드 리눅스

이 민 석*

● 목 차 ●

1. 서 론

내장형 리눅스는 Post-PC 시대에서 최적의 운영 체제가 될 것으로 기대되면서 현재 많은 회사와 연구 기관에 의해 개발, 구현되고 있다[1][2]. 서버 시장에서 리눅스의 성공은 Post-PC 시장에서의 성공을 바로 의미하지는 않는다. 리눅스는 전자 수첩, 무선 전화기 등 단순 단말기와 PDA, 스마트 폰으로 대표되는 정보 단말기의 중요한 구분 요인이 되는 소프트웨어의 유연성, 즉 새로운 응용 프로그램을 담을 수 있는 능력을 가지고 있지만 "작은 이동형 단말기에 필요한 모든 기능을 수용하는 운영 체제인가?" 라는 물음에, 있는 그대로의 리눅스는 "No" 라는 답을 할 수밖에 없다. 이 논문에서는 Post-PC 시대에서의 리눅스의 의미, 서버와 이동형 단말기의 차이에서 생기는 새로운 요구 사항을 살펴보고 이동형 단말기에 리눅스를 운영 체제로 사용하기 위하여 필요한 커널 수준의 기술들이 어떻게 구현되었는지 Tynux[3]에서의 개발 성과 위주로 설명한다.

* (주) 팜팜테크 CTO

2. Post-PC 운영 체제로서 리눅스가 가지는 장점

Linus Torvalds에 의하여 1991년 발표된 리눅스는 커널, 그래픽 사용자 인터페이스, 응용 프로그램, 개발 도구 등 모든 분야에서의 진화를 거듭하여 현재 인터넷 서버로 가장 많이 사용되고 있는 운영 체제로 자리 잡았다. 이러한 리눅스가 내장형 시스템 특히 Post-PC 단말기에서도 의미를 가지는 여러 가지 장점은 다음과 같다.

(1) 리눅스는 완전한 운영 체제이다. 리눅스 커널은 멀티 태스킹 환경, 페이지 기반의 메모리 시스템, 공유 라이브러리, TCP/IP 등 네트워크 프로토콜, 파일 시스템을 기본적으로 지원하고 있다. 특히 MMU에 기반한 메모리 관리 시스템은 쓰레드 모델로 실행되는 기존 실시간 운영 체제와는 달리 많은 응용 프로그램이 수행되는 정보 단말기에서 응용 프로그램들 사이 또는 커널과 응용 프로그램 사이의 메모리 영역 보호 기능을 수행한다. (2) 리눅스는 안정적이고 안전하다. 리눅스는 많은 개발자들에 의해 검토된 소스에 기반하기 때문에 배포 버전에 오류가 포함될 가능성이 다른 어떤 운영 체제보다 적다. (3) 리눅스의 모든 소스는 공개되어 있다.

리눅스는 소스와 내부 구조 및 모든 문서가 공개되어 있기 때문에, 사용자의 요구에 따라 커널 및 전 계층에 걸친 프로그램의 수정이 가능하며 커널 구조의 변경 및 재구성도 용이하다. (4) 개발 비용 및 사용료가 싸다. 리눅스의 커널 및 기본적인 응용 프로그램은 GPL 라이센스 정책[4]에 따라 무료로 제공된다. 또 소스와 많은 문서도 무료로 얻을 수 있기 때문에 개발 과정의 비용, 기술 지원을 받는 데 드는 비용, 제품에 부과되는 로열티를 최소화할 수 있다. (5) 개발 환경이 기본적으로 제공된다. 무료로 제공되는 리눅스에는 역시 무료로 사용할 수 있는 *GNU* 개발 도구가 포함되어 있다. (6) 리눅스는 모듈 단위로 설계되어 있다. 리눅스 커널은 사용자에게 필요한 기능과 하드웨어 장치에 따라 최적화된 구성으로 만들어져 메모리 사용량을 최소화할 수 있다. (7) 리눅스는 다양한 CPU, 다양한 컴퓨터 시스템에 이식될 수 있다. 리눅스는 현존하는 거의 모든 32bit CPU에 이식되어 있으며 새로운 CPU 나 컴퓨터에 쉽게 이식될 수 있는 구조를 가지고 있다. (8) 많은 장치 드라이버가 제공된다. 리눅스가 많이 사용되면서 다양한 하드웨어 장치에 대한 리눅스 드라이버가 하드웨어 제조 업체에 의하여 직접 또는 인터넷 상의 많은 개발자들에 의하여 속속 개발되고 있다. (9) 리눅스는 이미 많은 사용자가 사용하고 있고, 이미 많은 개발자가 있어 인력의 조달이 쉽고 결국 인건비가 대부분을 차지하는 프로젝트 비용을 절감할 수 있다.

3. 모바일 기기에서의 리눅스 기술

리눅스는 Post-PC 단말기와 같은 내장형 시스템에 가장 적합한 운영 체제로 부각되고 있지만 시장이 필요로 하는 기술적인 요구 사항들에 완전한 답을 가지고 있어야만 경쟁에서 살아남을 수 있다. 이동형 단말기의 운영 체제로서 가져야 하는 특성, 즉 시장에서 발생하는 주요 기술적 요구 사항은 다음과 같다.

커널 측면에서는 전력 관리 기능, 메모리 관리 기능, 플래쉬 파일 시스템, XIP (Execute In Place), 각종 장치 드라이버 지원 등이 있다. 미들웨어 측면에서는 그래픽 사용자 인터페이스, 보안 관련 솔루션, 원격 DB 참조 솔루션, Sync 관련 솔루션 등이 있고 마지막으로 응용 프로그램 측면에서는 기존 마이크로소프트사의 문서와의 호환성을 가지는 응용 프로그램들, 다양한 언어 지원, 멀티미디어 플레이어, 각종 플러그 인을 지원하는 브라우져, 개인 정보 관리, 게임 등이 있다. 아마도 나열된 대부분의 것들이 기존 데스크탑 리눅스에는 이미 존재하는 것들이지만 점유 메모리 크기, 화면 해상도 및 화면 크기가 다르기 때문에 수정되거나 새로 작성되어야 하는 경우가 적지 않다.

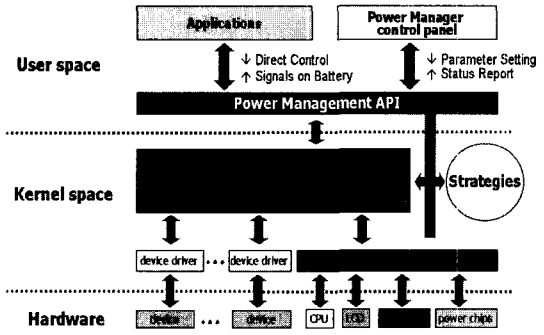
이 논문에서는 위 기술 가운데 핵심적이라고 할 수 있는 리눅스 커널 내부의 전력 관리와 메모리 관리 XIP 기법을 Tynux에서 어떻게 해결하고 있는지에 대하여 설명한다.

3.1 전력 관리

전력 관리는 전원을 배터리에 의존하는 휴대용 시스템에서 운영 체제가 가장 신경을 써야 하는 부분이다. 전력 관리의 주 목적은 사용 시간, 대기 시간을 포괄하는 배터리 사용 시간의 연장과 응용 프로그램들에 대한 전원 상태의 고지, "즉시 동작(Instant On)" 기능 등의 제공에 있다. 실제 전력 관리 구조의 가장 아래에는 하드웨어가 있으며 소프트웨어는 그 하드웨어를 이용하여 응용 프로그램의 동작에 따라 최소한의 전력을 사용할 수 있게 만드는 역할을 담당한다. 동적 전압 제어와 같은 경험적 알고리즘을 적용할 수 없는 경우, 일단 하드웨어가 주어지면 그 하드웨어의 모든 전력 관리 기능을 100% 이용하여 최소 전력이 소모되도록 만드는 것은 어떤 운영 체제 하에서도 가능하다. 따라서 운영 체제를 공급하는 측면에서 전력 관리 시

스템을 볼 때 필요한 것은 유연성의 확보이다. 실제의 Post-PC 단말기들은 표준 하드웨어에 의존하지 않기 때문에 전혀 새로운 하드웨어 장치, 통신 사업자가 제안하는 저전력 통신을 위한 프로토콜, 사용자의 단말기 사용 패턴 등 여러 가지 요인에 의하여 각기 다른 전력 관리 기법을 필요로 한다. 따라서 Post-PC 단말기의 다양한 응용에 빠른 시간 내에 최적으로 적응하기 위하여 운영 체제의 전력 관리 시스템의 구조는 최대한 유연하게 만들어져야 한다.

스마트 폰, PDA와 같은 휴대용 정보 단말기에 사용되는 리눅스를 위한 전력 관리 시스템의 구조는 (그림 1)과 같다.



(그림 1) Tynux의 전력 관리 시스템

위 전력 관리 시스템의 가장 큰 특징은 전력 관리의 작전(전력 소비를 줄이는 방법론 - 예를 들어 1분 간 사용자로부터 입력이 없는 경우 LCD와 LCD 조명을 끈다는 작전)과 전력 관리를 위한 제어(전력 소비를 줄이기 위한 기술적인 제어 - 실제 LCD를 끄는 API)를 분리하였다는 점이다. 이를 통하여 전력 소모를 최소화하기 위하여 일반적으로 사용하는 계층적 전력 공급 하드웨어 구조, 응용 프로그램의 장치 사용 패턴과 종속성을 전력 관리 작전과 분리해냄으로써 사용 전력을 줄이기 위한 새로운 하드웨어, 응용 프로그램이나 통신 프로토콜의 노력 등을 쉽게 운영 체제가 수용할 수 있게 된다. Tynux의 전력 관리의 핵심이 되는 전력 관리 모듈은 CPU나 각종 장치 하드웨어에 종속되지 않고 독립적으로 존재한다. 이 모듈은 시스템 내의 각종 장치 하드웨어의 동작을 제어/관찰하고, 전력 소모를 최소화하기 위하여 추가되는 새로운 전력 관리 작전에 필요한 인터페이스를 제공하여 Tynux 전력 관리 시스템이 새로운 환경에 쉽게 적응하도록 한다.

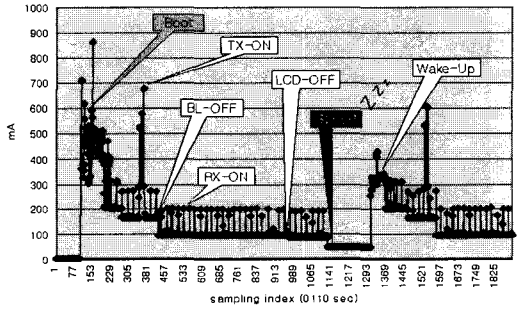
<표 1>은 PDA 형태의 스마트폰인 IMT-2000 프로토타입 단말기 (Intel StrongARM 206MHz, 640x480 4" TFT LCD, 32MB/32MB Flash/SDRAM, H/W Voice Codec, Bluetooth, MSM3000 CDMA 모듈)에서 대표적인 I/O 장치가 전력 소모에 기여하는 정도를 나타내고 있다. <표 2>는 같은 단말기에서 몇몇 응용 프로그램을 실행할 때 매우 단순한 수준의 전력 관리를 적용한 전후의 단말기 전체 전력 소모를 비교 표시한 것이다. 또 (그림 2)는 전화기 형태의 스마트 폰 (Intel StrongARM 206MHz, 240x160 2.5" STN LCD, 64MB/64MB Flash/SDRAM, Qualcomm MSM3100 CDMA Chipset)에서 부팅 후 시간대 별 전력 소모의 변화를 그래프로 표시한 것이다. 이 그래프에서 우리는 스마트 폰에서 전력 소모가 제어되는 것을 확인할 수 있다, 그림에서 주기적으로 나타나는 낮은 Peak는 수신을 위한 RF 장치의 동작 때 전류이고, 높은 Peak는 기지국으로의 데이터 송신 때의 소비 전류이다.

<표 1> PDA의 전류 소모

장치	소비 전류
LCD (On→⇒Off)	203mA
CCD (On⇒Off)	15mA
Audio (On⇒Off)	97mA
CPU (Run⇒Idle)	72mA
CPU (Run⇒Sleep)	124mA

<표 2> PDA에서의 전력 관리 효과 (mA)

응용프로그램	전력관리 Off	전력관리 On
None	778-798	LCD Off : 216
		LCD On : 474
Camera	801-813	650-662
Voice Recording	812-838	774-797



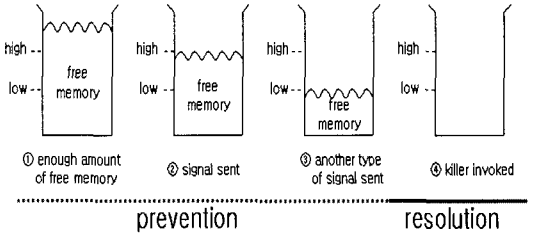
(그림 2) 스마트 폰에서의 전력 관리 효과

3.2 메모리 관리

메모리 보호 기능, 적은 메모리 사용량과 더불어 한정적인 메모리 자원을 효율적으로 사용할 수 있어야 한다는 것은 정보 단말기와 같은 단일 사용자 중심의 내장형 시스템에 사용되는 운영 체제가 갖춰야 할 조건 가운데 하나이다. 즉 메모리를 사용자 입장에서 가장 중요한 응용 프로그램의 실행에 유리하게 할당할 수 있도록 운영 체제가 관리해야 한다는 것이다.

메모리 가격이 낮아지면서 작은 단말기에도 점차 많은 양의 메모리를 사용하는 경향이 늘어나고 있다. 하지만, 서버와 같이 하드 디스크를 이용한 가상 메모리 시스템을 사용할 수 없는 내장형 시스템에서는, 호환성과 확장성이 유지되는 한, 운영 체제를 포함한 모든 소프트웨어가 사용하는 메모리의 양을 최소로 유지하는 것이 반드시 필요하다. 이는 커널, 라이브러리, 미들웨어 및 응용 프로그램의 최적화 및 파일 시스템의 압축 등을 통해 달성한다. 이 가운데 미들웨어 및 라이브러리의 최적화는 응용 프로그램 호환성, 확장성에 영향을 끼칠 수 있는 요인이 되므로 Post-PC 단말기에서는 좋은 선택이 아닌 경우가 많다. 응용 프로그램들끼리, 커널과 응용 프로그램 사이에서의 메모리 보호 기능은 다양한 응용 프로그램이 수행되는 Post-PC 단말기에서는 매우 중요한 요구 사항 가운데 하나이다. 리눅스는 이미 페이지 기반 메모리 보호 기능을 가지고 있기 때문에, 기존의 실시간 운영 체제에서 한 응용 프로그램의 오류가 시스템 전체의 정지 또는 오류로 이어지는 심각한 문제에서는 자유롭다.

Tynux에서는, 언제나 단일 사용자만 존재하는 개인용 단말기에서 가장 중요하다고 사료되는 "현재 사용 중인 응용 프로그램" (foreground process)이 메모리가 부족해서 문제를 겪지 않도록 돕기 위한 메모리 관리 기법을 제공하고 있다. 이 기법은 응용 프로그램이나 라이브러리들이 성능 향상 등의 이유로 사용하고 있는 버퍼 또는 캐쉬 영역이 스스로의 의지에 의해 안전하게 반환이 가능하다는 점에 기반하고 있다. (그림 3)에서와 같이 Tynux의 메모리 관리는 두 단계로 이루어진다. 첫 번째 단계는 메모리 부족을 막는 메모리 부족 방지 (prevention) 단계이고, 두 번째 단계는 최종적으로 더 이상 메모리가 남아 있지 않은 경우 빈 공간을 확보하는 메모리 부족 해결 (resolution) 단계이다. 방지 단계에서 메모리 관리 모듈은 시스템의 남은 메모리 양을 관찰하고 있다가 그 양이 어느 수준 (high-watermark) 이하로 떨어지면 응용 프로그램들에 신호를 보내 현재 사용하고 있는 메모리 가운데 불요불급한 것들을 응용 프로그램들이 자진 반납하도록 한다. 또 더 낮은 수준 (low-watermark) 이하



(그림 3) Tynux의 메모리 관리 개요

로 떨어지면 라이브러리 수준에서 남아 있던 안 쓰는 메모리까지 모두 반환하도록 한다.

메모리 부족 방지를 위한 노력에도 불구하고 결국 메모리가 바닥이 나면, 할 수 없이 메모리 부족 해결 단계로 들어가 현재 실행되고 있는 응용 프로그램 가운데 하나 또는 그 이상의 실행을 중단시켜 필요한 양의 메모리를 확보하게 된다. 이 작업을 수행하는 킬러 프로그램은 미리 정의된 우선 순위에 의해 응용 프로그램을 중단시키거나 사용자가 임의로 중단시킬 프로그램을 선택하도록 만들어져 있다.

Tynux의 메모리 관리 시스템은 커널의 스케줄러와 페이지 캐쉬 구동 메카니즘, 라이브러리의 수정을 통하여 구현되었으며, 응용 프로그램은 메모리 관리를 위해 추가된 시그널 (GUI를 이용하는 응용 프로그램의 경우 메모리 상태 이벤트)에 대하여 스스로 정한 기준에 의해 자신이 할당받은 버퍼, 캐쉬 영역의 일부를 릴리즈 함으로써 동작한다.

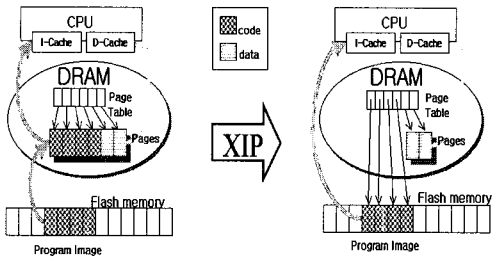
리눅스 커널에 도입된 OOM (out-of-memory) Killer 방식은 메모리 부족을 미리 방지하는 기능이 없으며, 메모리 부족 사태가 발생하면 사후 처방으로 임의의 프로세스를 중단시키며 최근에 이를 조금 보완한 버전이 구현되고 있다. 따라서 Tynux에 사용된 방법은 성능과 사용자 입장에서의 각 프로세스의 중요성 등이 고려된 방법이라고 할 수 있다.

3.3 XIP 기술과 플래쉬 파일 시스템의 연동

이 절에서는 내장형 시스템에서 보편적으로 사용되는 XIP 기술을 소개하고, 실제 XIP 기술이 사용될 때 플래쉬 파일 시스템과의 연동에 의해 생기는 문제를 해결하는 방법을 살펴본다.

3.3.1 XIP 기술

내장형 시스템에 필요한 또 하나의 기술은 XIP (Execute In Place) 기능이다. 리눅스를 비롯한 기존의 범용 운영 체제들이 하드디스크에 있는 실행 이미지를 DRAM에 읽어 들인 뒤 실행되는 것과는 달리, 내장형 시스템에서는 실행 이미지가 이미 CPU의 주소 공간에 존재하는 메모리 (플래쉬 메모리, 마스크 롬 또는 DRAM) 위에 존재하기 때문에, 파일을 DRAM에 다시 로드하지 않고 그대로 실행할 수 있으며 이러한 기술을 XIP라고 한다. 최근에 만들어지는 대부분의 내장형 시스템에서는 프로그램 이미지 저장을 위한 비휘발성 메모리로 NOR 형태의 플래쉬 메모리를 사용하기 때문에 XIP에 관한 논의는 플래쉬 메모리를 대상으로 한정한다. (그림 4)는 XIP가 이루어지는 과정을 보여 주고 있다. 그림에서처럼 Tynux에서의 XIP는 코드 페이지들을 DRAM이 아닌 실행 파일이 저장된 플래쉬 메모리 (엄밀히 이야기하면 플래쉬 파일 시스템의 코드 페이지 블럭)에 직접 매핑함으로써 이루어진다. 따라서 XIP가 되기 위한 최소한의 요건은 실행 파일이 플래쉬 메모리 상에 페이지에 맞추어 정렬 (align)되어 있어야 한다는 것이다.



(그림 4) XIP 의 동작 원리

XIP를 사용함으로써 얻는 이점들은 다음과 같다 : 파일 시스템에서 DRAM으로의 실행 이미지 로드가 없어짐으로 인한 DRAM 요구량 감소와 프로그램 로딩 시간의 개선, 페이지 캐싱이 없어짐으로 인한 문맥 교환 속도 향상, 전력 소비가 적은 플래쉬 메모리 사용으로 인한 전력 소비 감소 효과, 플래쉬 파일 시스템 상의 파일의 메모리 매핑을 통한 데이터 파일 참조 성능 개선, 하나의 NOR-플래쉬 메모리 사용으로 인한 공간 절약 등이 있다.

반면에 XIP를 사용해서 발생할 수 있는 다음과 단점들도 있다 : 플래쉬 메모리의 낮은 메모리 대역폭으로 인한 성능 저하 (이 단점은 대부분 임베디드 CPU에 내장된 On-Chip 캐쉬에 의해 많이 상쇄된다. 측정 결과도 이 부분에 의한 성능 저하는 문제가 되지 않는 수준이며, 200Mhz 이상의 높은 클럭 주파수를 가지고 실행되는 휴대용 정보 단말기에서 이 대역폭 차이에 의한 성능 저하는 사용자에게 거의 느껴지지 않는다. 이 낮은 대역폭 문제는 CPU의 LCD refresh 관련 회로 구조에 따라 큰 해상도의 LCD에서 플리커링 현상을 일으킬 수 있다.), 상대적으로 값이 비싼 NOR-플래쉬 메모리의 사용으로 인한 제품 가격의 상대적 상승 (이 단점은 DRAM의 급속한 가격 하락과 NAND-플래쉬의 가격 대 용량 비의 개선으로 인하여 심화되고 있다.), 플래쉬 메모리 파일 시스템과의 연동에 의해 생기는 파일 쓰기 참조의 지연 (이 단점을 해결하기 위한 방법이 아래 제시되어 있다.)

위와 같은 단점에도 불구하고 XIP 기술은 제품 가격, 시스템의 성능, 전력 소모량, 제품의 크기 등에 관한 트레이드-오프를 제공하기 때문에 다수의 응용 프로그램이 실행되어 메모리가 많이 필요한 휴대용 정보 가전에 중요한 기술이다.

3.3.2 플래쉬 파일 시스템과 XIP

플래쉬 메모리는 다음과 같은 특이한 특성을 갖는다[5]: (1) 비교적 큰 크기의 블록 단위로만 지울 수 있다. (2) 같은 블록에 대한 쓰기, 지우기 동작이 반복되면 그 블록이 닳아 결국 에러로 인하여 그 블록을 사용할 수 없게 된다 (특히 NAND 플래쉬의 경우 이 문제의 해결이 중요하다). (3) 쓰기 동작에는 읽기 동작보다 비교적 긴 시간이 소요되며 지우기 동작에는 무시할 수 없는 수준의 상당히 긴 시간이 소요된다. (4) 플래쉬에 쓰기나 지우기가 진행되는 동안 읽기 동작이 금지된다.

따라서 XIP를 쓰기가 가능한 플래쉬 화일 시스템과 함께 운영하는 경우에 위의 플래쉬 메모리의 특성[5]을 고려해야한다. 위 특성 가운데 앞의 두 개는 플래쉬 파일 시스템[6]에서 해결하고 있으며 뒤의 두 가지 특성이 XIP와 연계되어 문제를 일으킨다. 이 문제는 플래쉬에 쓰기 또는 지우기 진행되는 동안 플래쉬 메모리를 읽지 못하게 되므로 어떤 프로그램도 실행할 수 없는 상태를 만드는 문제이다. 이 문제에 대한 가장 간단한 답은 플래쉬에 쓰기나 지우기 동작이 진행되는 동안 아무 일도 하지 않고 대기하는 것이다. 하지만 이런 해결책은 과도한 대기 시간을 발생시켜 이 대기 시간동안 전화 걸기, 받기 등과 같은 중요한 프로그램이 중단되는 사태를 야기할 수 있어 시스템의 가용성을 낮추게 된다. 이 문제 때문에 마이크로 소프트의 Pocket PC 운영 체제와 같은 대부분의 정보 단말기 운영 체제에서는 데이터 저장소로 플래쉬 메모리가 아닌 DRAM을 사용하며, 전원이 꺼지면 데이터도 지워지는 DRAM의 휘발성을 PC와의 동기화로 해결하고 있다. 이 방법은 비교적 큰 용량의 배터리로 운용되며 충전과 동기화를 비교적 자주 할 수 있는 환경에서는 유효하나 그렇지 않은 경우에는 사용할 수 없다.

Tynux에서는 이 보다는 좀더 현실적인 대안을 제시하고 있다. Tynux에서는 대부분의 NOR-플래쉬 메모리가 제공하는 쓰기/지우기의 비동기적인 정지 및 재시작 기능을 십분 활용하여 스케쥴러가 현재 CPU의 상태와 실행할 프로그램을 고려하여 동적으로 플래쉬 메모리의 쓰기/지우기를 제어한다. 이 제어를 통하여 프로그램은 플래쉬에 쓰기 또는 지우기 요구가 있거나 진행되고 있는 상태에서도 실행이 계속된다. 물리적으로 플래쉬에 대한 쓰기/지우기 시간과 읽기 시간을 오버랩 시킬 수 있는 방법은 없기 때문에 스케줄러가 이용할 수 있는 시간은 CPU가 현재 실행하는 코드가 확실히 DRAM이거나 Run 큐에 스케줄할 프로세스가 없는 비플래쉬 참조 시간을 이용한다. 대량의 쓰기 요구

가 플래쉬 파일 시스템에 몰리는 경우에는 앞서 언급한 비플래쉬 참조 시간이 상대적으로 너무 짧기 때문에 플래쉬 파일 시스템의 블록 삭제 요구 큐를 검사하여 어떤 한계를 넘어서면 시스템의 모든 코드 페이지를 조작하여 XIP를 포기함으로써 플래쉬 메모리 지우기와 프로그램 실행을 동시에 수행한다. 즉 이 순간 이후부터는 플래쉬 메모리의 코드도 DRAM으로 on-demand로 복사되어 실행되며, 플래쉬 메모리에 대한 지우기 요구가 다시 줄어들면 XIP를 재개한다. 이 과정에서도 메모리 관리 기법에서 제공되는 페이지 캐쉬 제어에 의하여 DRAM 요구량이 폭발적으로 증가하지는 않는다.

4. 요약 및 남은 과제

리눅스는 Post-PC 시대에서 최적의 운영 체제로 부각되고 있다. 하지만 서버 시스템에서 주로 사용되고 검증된 리눅스가 이동형 정보 단말기에 적용되기 위하여는 기능적 측면에서의 보완이 많이 필요하다. 이 글에서는 전력 관리, 메모리 관리, 파일 시스템, XIP 기능 등 이동형 내장 기기에 사용될 리눅스 커널의 기능적 요구 사항에 대하여 살펴보고 Tynux가 제시하는 해결책을 살펴보았다.

많은 임베디드 리눅스 관련 연구와 개발에도 불구하고 현재까지는 리눅스를 휴대용 정보 단말기에 내장하는 일은 실험적 성격으로 이루어져 왔다. 실제 시장에의 성공적인 진입을 위해서는 다양한 단말기용 응용 프로그램 요구에 대한 준비가 선행되어야 하며, 다음과 같은 기술적 요소들이 더 충족되어야 할 것으로 보인다.

- 작은 크기의 범용 응용 프로그램의 개발, 인터넷, 멀티미디어, 오피스웨어
- 네트워크에 연결된 상태로 존재하는 단말기에서의 시스템 자체 보안, 다운로드되는 프로그램의 우발적/악의적인 위협에 대한 보안
- 이전의 유선 네트워크와는 전혀 다른 특성을 보이는 무선 네트워크가 가진 여러 문제들에 대하여 우아하게 대응을 하는 TCP/IP 프로토콜의 구현
- 플래쉬 메모리, 플래쉬 메모리 파일 시스템을 고려한 효율적인 데이터베이스 관리 기법
- 실시간 시스템에서와 같은 예측 가능성과 고정 우선 순위를 가지지 않는 비 실시간 실행 환경에서 전압 및 클럭 주파수 제어를 통한 저전력 시스템을 만들기 위한 효율적인 경험적 알고리즘 개발
- 일반적으로 낮은 메모리 대역폭을 보이는 휴대용 기기를 위한 커널의 성능 향상
- RAD (Rapid Application Development) 툴을 포함하는 완벽한 통합 개발 환경

참고문헌

[1] "Presentation: The State of Embedded Linux", LinuxDevice.com Homepage, http://www.linuxdevices.com/articles/AT2113794413.html, July, 2000.

[2] 정갑주, 이민석, 최건, "내장형 리눅스", 정보과학회지, Vol.18, No.2, 2000년 2월.

[3] 팜팜테크(주), http://www.palmpalm.com

[4] Free Software Foundation, "GNU General Public License", http://www.gnu.org/copyleft/gpl.html, 1991

[5] Intel Flash Memory Data sheets, http://developer.intel.com/design/flash/datashts/index.htm

[6] Cliff Brake and Jeff Sutherland, "Flash File Systems for Embedded Linux Systems", Embedded Linux Journal, Issue 04, July/August 2001

저자약력

이 민 석

1986년 서울대학교 컴퓨터공학과(공학사)
1988년 서울대학교 대학원 컴퓨터공학과(공학 석사)
1995년 서울대학교 대학원 컴퓨터공학과(공학 박사)
1990년-1993년 (주)이미지 시스템, 미디어연구소 소장
1998년-1999년 노스캐롤라이나 주립대 컴퓨터공학과 박사 후 연구원
1995년-현재 한성대학교 컴퓨터공학부 부교수
1998년-현재 (주) 팜팜테크 CTO
관심 분야 : 내장형 시스템, 실시간 시스템, 컴퓨터 구조
e-mail : mslee@hansung.ac.kr