임베디드 시스템을 위한 리눅스의 빠른 부팅 기법

박성호* · 신광무** · 김영주***

A Fast Booting Scheme of Linux for Embedded System

Seongho Park · Kwangmu Shine · Youngju Kim

이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음

요 약

안정성, 유연성, 오픈 소스, 다양한 하드웨어 플랫폼 지원, 검증된 네트워크 등의 장점을 가지고 있지만 데스크탑 시스템이상의 높은 성능을 가진 시스템에 최적화되어 있는 리눅스 운영체제를 스마트 임베디드 시스템에 탑재하기 위해서는 수 십초가 걸리는 부팅시간을 단축하여야 하는 문제점을 해결하여야 한다.

본 논문은 임베디드 환경에서 범용 운영체제인 리눅스를 활용하여 빠른 부팅 기법을 제안하고 프로토타입의 구현을 통해 성능을 평가하였다. 특히 부팅시간에 영향을 미치는 부트로더, 커널, 루트 파일시스템 등의 각 구성요소를 최적화함으로써 부팅 시간을 최소화하였다. 그리고 구현 결과를 HBE-EMPOS II 기준으로 실험하였으며, 실험결과 부팅시간이 최초 28초에서 11초로 감소되는 성능 향상을 가져왔다.

ABSTRACT

Linux has peculiarities of safety, flexibility, and open source. and Linux supports various hardware platforms. But because general Linux was designed for high performance H/W, Linux has several points to support for embedded system with limited resource. Specially, a main point is booting time that is spent to tens of second.

In this paper, we purpose a fast booting scheme of Linux for Embedded System and estimate a performance of scheme purposed through realized prototype. Specially, to reduce booting time in environment of embedded system, we optimize boot loader, kernel and root file system. As a result, boot time reduce 28 second to 11 second in HBE-EMPOS II.

키워드

Booting Time, Embedded System, Embedded Linux, Separated Loading

I. 서 론

초기의 임베디드 시스템은 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자 제어 시스템으로 아주 간단한 제어 기능을 수행하는 8비트 MCU와 같은 프로세서로 개발되어 운용되었

* 부산대학교 정보전산원

** 부산대학교 컴퓨터공학과

*** 신라대학교 컴퓨터공학과(교신저자)

접수일자 : 2006. 8. 17

으며, 특별한 운영체제를 요구하지 않았다. 이에 반해, 오늘날의 임베디드 시스템은 사용자의 요구에 따라 많은 복잡한 기능을 수행하여야 하며, 특히, 휴대 전화, 인터넷 셋톱박스, 네트워크 장비, 산업기기 등은 거의 개인용 컴퓨터 수준의 복잡한 기능을 수행하도록 요구되어지고 있다. 즉, 최근의 임베디드 시스템은 간단한 제어 기능만을 수행하는 펌웨어 수준이 아니라 복잡하고 다양한 기능을 수행하는 인텔리전트한 스마트 시스템 수준의 성능을 요구하고 있으며, 이를 위해서는 멀티프로세싱, 네트워킹 등의 고급 기능이 제공되는 범용 운영체제 수준의 성능을 가진 운영체제가 탑재되어야 한다. 현재 운용되고 있는 임베디드 시스템은 WinCE, Pocket PC, Linux 등의 범용 운영체제를 다수가 채용하고 있으며, TRON, BlueCat POS와 같이 특정 응용 시스템에 특화된 운영체제를 적용한 시스템도 다수 운용되고 있다. 그러나 WinCE나 Pocket PC의 경우 임베디드 시스템 생산업체에서는 운영체제 개발을 위한 투자 부담이 없으나 장기적으로 높은 라이센스 비용에 따른 임베디드 시스템의 비용 증가 문제와 시스템 성능 향상을 위한 즉각적인 운영체제 지원 등에 많은 어려움이 발생할 것으로 예상된다. 또한 TRON, BlueCat POS 등은 펌웨어 수준의 운영체제로서 개발 및 운영 비용은 절감될 수 있으나 인텔리전트한 수준의 기능을 제공하지 못하고 있다.

리눅스는 안정성, 유연성, 오픈 소스, 다양한 하드웨어 플랫폼 지원, 검증된 네트워크 등을 장점으로 스마트 임베디드 시스템의 운영체제로 많은 관심을 받고 있다. 그러나 기존의 리눅스 시스템은 데스크 탑 시스템 등과 같이 성능이 높은 시스템에 최적화 되어있기 때문에 자원 제약이 많은 임베디드 시스템 환경에 적용하기 위해서는 해결해야할 몇 가지 문제점을 가지고 있다. 특히 기존의 리눅스 시스템은 20~40초의 긴 부팅 시간이 요구됨으로써 이전에 PnP(Plug-and-Play) 방식으로 사용하는 전통적인 가전제품이나 산업기계용 임베디드 시스템에 적용할 경우에는 사용자에게 많은 불편을 발생시킬 수 있을 뿐만 아니라, 제품의 오동작으로부터 빠른 복구를 위해 빠른 부팅을 필요로 하는 장비의 중요한 요구를 만족시킬 수 없다.

본 논문은 리눅스를 스마트 임베디드 시스템에 적용시키기 위하여 부팅시에 적재되는 부트로더, 커널, 루트파일시스템과 같은 부팅 요소의 경량화를 통한 빠른 부팅 기법을 제안한다. 본 논문에서는 리눅스 커널의 최적화 작업을 통해 임베디드 시스템의 하드웨어 및 소프트웨어 요구에 적합하도록 커널을 최적화시켰으며, 본 논문에서 제안한 분리 적재(Separated Loading) 방식에 근간하여 루트파일시스템과 부트로더 모듈을 기능별로 분리하여 경량화를 수행함으로써 빠른 부팅을 지원한다. 테스트 임베디드 장비를 통한 성능 실험 결과, 부팅시간이 최초 28초에서 11초로 감소시키는 성과를 얻었다.

본 논문에서의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 기존의 연구를 살펴보고, 3장에서는 빠른 부팅을 위한 임베디드 리눅스 경량화 설계를 제안한다. 그리고 4장에서 테스트 임베디드 장비를 이용한 실험 결과를 분석하여 제안된 빠른 부팅 기법의 성능을 평가한다. 그리고 마지막으로 5장에서 결론 및 향후 연구 과제를 기술한다.

II. 관련 연구

임베디드 시스템에 기존의 리눅스 운영체제를 적용하기 위해서는 커널의 경량화, 부팅 시간의 최소화 등의 문제를 해결하여야 한다. 본 장에서는 임베디드 시스템 환경에서 리눅스 운영체제의 빠른 부팅을 지원하기 위하여 다양한 시스템에서 적용된 빠른 부팅(Fast Booting) 및 빠른 로딩(Fast Loading)에 관한 선행 연구를 살펴보고, 임베디드 시스템 환경에서 리눅스 운영체제 부팅과 관련된 시간 지연 요소 등에 대하여 살펴본다.

2.1. 빠른 부팅 기법

임베디드 시스템에서 빠른 부팅 및 로딩을 구현하기 위해 XIP(eXecute-In-Place) 기법에 관한 연구가 다수 진행되었다[1]. XIP 기법은 실행 코드를 RAM으로 복사하지 않고 플래시 메모리 영역에서 바로 실행하는 기법으로서, 플래시 메모리를 보조 저장 장치로 채택하고 있는 임베디드 시스템 환경에서 구현이 가능하여 많은 관심을 받고 있다. 그러나 표 1에서 살펴보듯이 플래시 메모리는 충분히 빠른 읽기 접근 시간을 제공하고 있지만 SDRAM에 비해 10배 정도 느린 접근 속도를 가지고 있다. 이는 실행 빈도가 낮은 소프트웨어에 적용할 경우에 시스템 전체 성능에 많은 영향을 미치지는 않지만, 운영체제 모듈과 같이 실행 빈도가 높은 코드를 플래시 메모리에 위치시켜 실행하는 운영체제 XIP 기법은 시스템 전체 성능을 저하시키는 문제점을 발생시킬 수 있다. 그러므로 최근에는 XIP 기

법을 통해 부팅 시간 감소보다는 주기억장치(RAM) 사용량을 줄이기 위한 기법으로 연구되고 있다[2,3,4,5,6].

표 1. 플래시 메모리 vs. DRAM의 읽기 접근 시간 비교
Table 1. The comparison of Read access time between flash memory and DRAM

특징	플래시 intel 28F640J3A	SDRAM	DRAM		
			FPM	EDO	BEDO
읽기 접근 시간	100~150 ns	10~15 ns	25~35 ns	20~30 ns	15~20 ns

출처:Intel, EDN

또한 다양한 시스템 환경에서 리눅스 운영체제의 부팅 시간을 감소시키기 위하여 Calibration 지연 감소 기법, 지연된 IDE Probing 기법, 「quiet」 옵션 등 내부 옵션을 이용한 기법들이 제시되었다[7,8]. 「loops_per_jiffy」 값을 적절히 조절하여 Calibration 지연을 감소시키는 기법 및 커널 명령어 라인에 「quiet」 옵션을 주어 시리얼 인터페이스를 통해 출력되는 부팅 메시지를 제거하는 기법은 수 ms 정도의 매우 미약한 부팅 시간 감소 효과를 거두고 있다. 또한 지연된 IDE(Integrated Device Electronics) Probing 기법은 하드 디스크 등 보조 기억장치의 고용량화로 장치 초기화 시간이 길어짐에 따라 하드디스크 장치 초기화와 IDE 인터페이스 장치 초기화 과정을 고려하여 IDE 장치 검사를 적절하게 지연시킴으로써 전체 부팅에 소요되는 시간을 단축시키는 기법이다. 그러나 물리적 초기화 과정이 필요 없는 플래시 메모리를 주로 사용하는 임베디드 시스템에는 이 기법이 적용 대상이 될 수 없다.

2.2. 임베디드 시스템에서의 운영체제 부팅

리눅스 운영체제의 부팅은 부트로더(Bootloader) 실행과 커널 부팅 단계로 이루어진다. 특히, 부트로더는 수십 KB 에서 200KB 이하의 크기를 가지며, 하드 디스크 또는 플래시 메모리의 특정 영역에 저장되어 있다. 그리고 사용자가 시스템 사용을 위해 파워를 켜면 시스템은 부트로더를 가장 먼저 메모리로 적재하여 부트로더를 실행한다. 그림 1은 리눅스 운영체제를 탑재하고 있는 테스트 임베디드 장치인 한백 HBE-EMPOS II 에서의 부트로더의 실행 과정을 예로 보여준다. 일반적으로 부트로더는 시리얼 포트, 기타 I/O 디바이스, 네트워크 장치 등의 하드웨어의 초기화, 부트로더 커맨드 수행, 커널과 루트파일시스템 메모리 적재 및 압축 해제 등의 기능을 단계별로 수행한다.

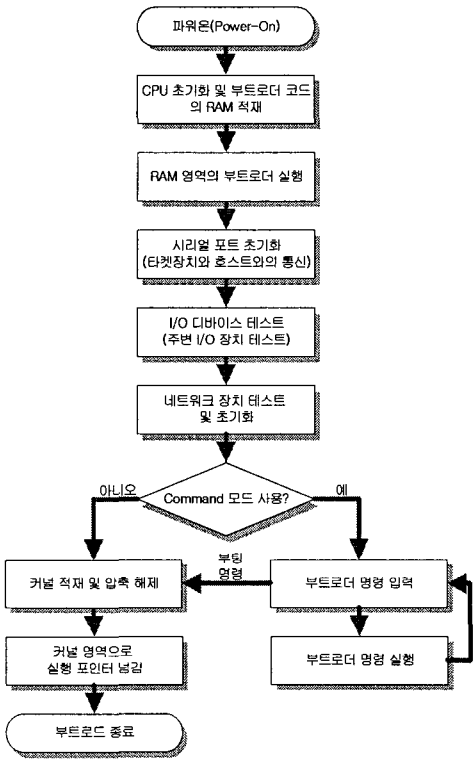


그림 1. 테스트 임베디드 장치에서의 부트로더 실행 과정
Fig. 1. The execution course of boot loader on embeded system for test

부트로더 단계가 성공적으로 수행되면, 커널 부팅 단계로 넘어간다. 커널 부팅 단계에서는 컴퓨터 하드웨어를 운영하는데 필요한 소프트웨어를 초기화한다. 그리고 루트파일시스템에 탑재된 환경 파일을 검색하여 시스템에 사용에 최적화될 수 있도록 파일시스템 마운트 및 각종 데몬(Demon)을 적재하고 초기화 한다[9, 10]. 테스트 임베디드 장비에서는 리눅스 커널 2.4.19을 기준으로 그림 2와 같은 부팅 과정이 이루어지는데, 부팅이 완료되는 시간은 약 30초 내외로 일반 임베디드 시스템에 적용되기에는 부적합하다.

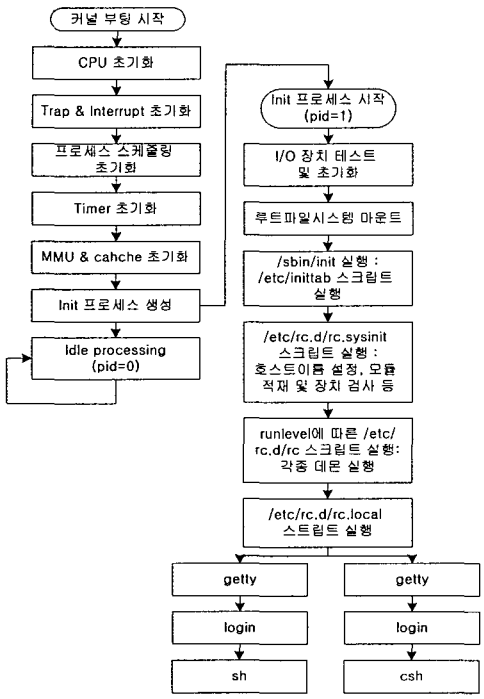


그림 2. 테스트 임베디드 장비에서의 커널 부팅 과정
Fig. 2. The course of kernel booting on embeded system for test

2.3. 리눅스 루트 파일시스템의 구조

일반적으로 임베디드 시스템에서는 시스템이 부팅될 때에 루트파일시스템을 휘발성이 없는 기억장치에서 휘발성 메모리인 RAM으로 복사하여 재구성한다[9, 10]. 이는 루트파일시스템이 읽기전용 파일시스템이고 리눅스 커널이 실행하는 동안 빈번하게 접근하기 때문에 시스템의 전체 실행 속도를 향상시키기 위함이다. 리눅스 루트파일시스템의 최상위 레벨의 구조는 표 2와 같은 구조를 가지고 있다. 그러나 범용 시스템을 위한 리눅스 루트파일시스템은 다양한 기능을 지원하기 위하여 매우 많은 기능을 루트파일시스템에 포함하고 있어 메모리 적재 및 재구성에 많은 시간을 소비함으로써 임베디드 시스템의 부팅 시간을 증가시키는 요인이 되고 있다.

표 2. 루트 파일시스템 최상위 레벨 구조
Table 2. The first level architecture of root file system

디렉토리	내용
bin	필수적인 사용자 명령 바이너리
boot	부트로더가 사용하는 정적 파일
dev	장치 파일과 특수 파일
etc	시스템 시작 파일을 포함한 시스템 환경 설정 파일
home	사용자 홈 디렉토리
lib	C 라이브러리나 커널 모듈 등의 필수 라이브러리
mnt	일시적으로 마운트된 파일시스템에 대한 마운트 지정
opt	부가적인 소프트웨어 패키지
proc	커널과 프로세스 정보를 위한 가상 파일 시스템
root	루트 사용자의 홈 디렉토리
sbin	필수적인 시스템 관리 바이너리
tmp	임시파일
usr	X 서버를 포함해서 사용자에게 유용한 대부분의 응용 프로그램과 문서를 포함하는 서브 디렉토리
var	데몬과 유틸리티에 의해 저장되는 변수 데이터 디렉토리

III. 임베디드 리눅스의 빠른 부팅 기법 설계

본 논문에서는 임베디드 리눅스의 빠른 부팅을 지원하기 위하여 리눅스 경량화와 분리 적재 기법을 제안한다. 경량화된 리눅스는 RAM으로의 복사 시간, 압축해제 시간, 프로세스 초기화 시간을 단축하는데, 리눅스의 경량화는 크게 부트로더 (Bootloader)의 경량화, 리눅스 커널 (Kernel)의 경량화, 루트파일시스템 (Root Filesystem)의 경량화로 나누어 수행하였다.

3.1. 부팅 최적화 메커니즘

본 논문에서는 부팅 메커니즘을 최적화하기 위하여 다음의 두 가지 기법을 적용하였다.

첫째, 기능 최적화를 통한 모듈 경량화 - 임베디드 장치의 하드웨어 구성과 응용 목적 등을 고려하여 부팅 요소를 구성하는 모듈 중에서 불필요한 부분을 제거함으로써 기능적으로 부팅 요소를 최적화시킨다.

둘째, 분리 적재(Separated Loading) - 정상적인 부팅 과정에 필요한 요소만을 분리하여 우선 적재하고 부가적인 요소에 대해서는 필요할 때에 적재하도록 함으로써 적재에 따른 부팅 시간을 단축한다.

상기의 두 가지 기법을 적용함으로써 그림 3과 같이 부트로더 단계에서 정상적인 부팅에 필요한 모듈만을 포함하고 있는 부팅 최소 모듈, 포팅 대상 보드를 위해 최적화된 커널 이미지, 커널이 빈번히 사용하는 기능만을 포함한 기본 루트 파일시스템 만을 RAM에 적재하여 리눅스 부팅 과정을 최적화하였는데, 이러한 적재 과정은 RAM으로의 복사 및 압축해제에 따른 소요시간을 최적화한다.

또한 추가적으로 부트로더 단계에서 불필요한 초기화 과정을 제거하고 시스템 초기화(RC-Run Command) 스크립트에서 불필요한 초기화 과정을 제거하여 부팅시간을 단축하였다.

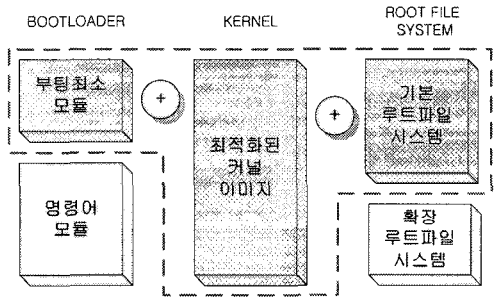


그림 3. 최적화 메커니즘 통해 램에 복사되는 S/W
Fig. 3. The software copied to RAM by optimized mechanism

3.2. 부트로더 경량화

임베디드 시스템의 부트로더 구조는 그림 1에서 보듯이 각종 디바이스 초기화 및 테스트를 거친 후 자동 부팅 여부를 결정하고 그 선택에 따라 커널 영역으로 실행 포인터를 넘기거나 명령어 모듈을 실행한다. 명령어 모듈은 하드웨어의 configuration을 변경하거나, 시스템 재설치 등을 수행하는 모듈로서 정상적인 부팅 과정에는 관여하지 않는다. 그러므로 본 논문에서는 명령어 모듈을 부분을 하나의 독립된 모듈로 재구성하여 정상적인 부팅과정에서는 RAM으로 복사되지 않도록 함으로써 부팅시간을 단축하였다. 즉, 독립된 명령어 모듈은 인터럽트가 발생할 경우에만 메모리로 복사하여 구동함으로써 실제적인 부팅 시간의 감소를 가져오도록 그림 4와 같이 설계하여 구현하였다.

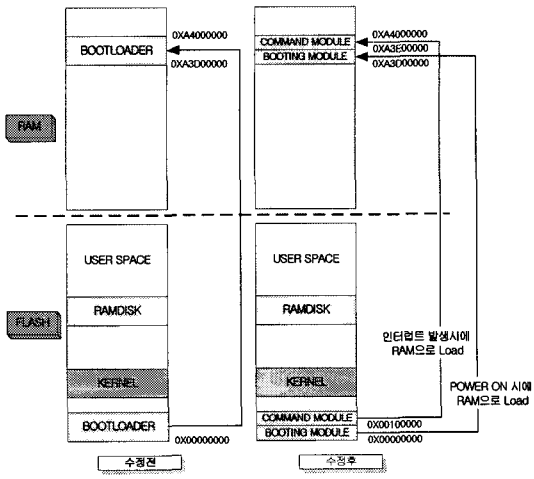


그림 4. 부트로더의 메모리 맵 구조
Fig. 4. The architecture of boot loader's memory map

3.3. 리눅스 커널 경량화

리눅스 커널은 범용 목적으로 운영되기 때문에 운영체제가 포팅될 임베디드 시스템의 하드웨어 구성과 응용 목적에 적합하지 않은 많은 기능을 커널에 포함하고 있다. 이와 같이 해당 임베디드 시스템에 불필요한 커널 코드는 커널의 크기를 증가시켜 플래시 메모리에서 RAM으로 복사하는 시간 및 압축해제 시간의 증가를 가져온다. 본 논문에서는 임베디드 시스템에 불필요한 기능을 제거하여 커널 크기를 최소화함으로써 임베디드 시스템의 부팅 시간을 줄인다. 또한, 이러한 과정을 통해 불필요하게 커널에 포함된 모듈이 적재되는 시간을 절약할 수 있다.

본 논문에서는 커널 크기의 최적화 방법으로서 커널 configuration 최적화를 통한 커널 크기 최적화를 수행하였다. 즉, 일반적으로 임베디드 시스템이 구동하는데 꼭 필요한 메모리 및 프로세스의 관리, 인터럽트 및 시그널 관리, 시스템 콜 등의 커널 코어(Core) 기능 및 다양한 서비스를 위한 네트워크, MTD, 파일시스템 등 임베디드 시스템의 성능에 영향을 미치지 않는 부분을 제외하여 최적의 커널이미지를 생성하도록 커널 configuration을 최적화하였다. 이때 커널의 각 기능 간의 의존성을 해치지 않는 범위 내에서 configuration 최적화가 이루어져야 한다. 표 3은 커널 configuration 최적화 과정에서 옵션 선택 최적화의 예이며, 임베디드 테스트 장비인 HBE-EMPOS II 보드에서 제공하는 기본 커널 configuration 에서 필요하지 않은 옵션을 제거하는 방식을 취하였다.

3.4. 루트파일시스템 경량화

리눅스는 루트파일시스템을 통해 하드디스크나 I/O 디바이스 등에 접근하는 구조를 가지고 있다. 임베디드 시스템에서는 루트파일시스템에 매우 빈번한 접근이 발생할 뿐만 아니라 읽기전용 파일시스템인 특성 등을 고려하여 시스템의 성능을 향상시키기 위해 RAM에 루트파일시스템을 구축하는 구조를 가지고 있다. 이 루트파일시스템도 커널과 동일하게 압축된 상태로 플래시 메모리에 저장되어 있으며 부팅 시에 RAM으로 복사되어 압축을 해제하는 과정을 거친다. 그러므로 루트파일시스템의 경량화는 리눅스 부팅 속도를 향상시킬 수 있다.

루트파일시스템 경량화 과정에 사용하는 최적화 방법은 루트파일시스템을 부팅 시에 커널이 필요로 하는 기본 영역과 부팅 완료 후에 애플리케이션 프로그램이 필요로 하는 확장 영역으로 구분하여 재구성한다. 즉, 커널의 수행 과정에서 필수적인 라이브러리와 데이터는 기본 루트파일시스템에 포함되도록 구성하고 나머지 라이브러리 및 데이터를 담은 확장 루트파일시스템으로 분리되도록 구성한다. 이렇게 분리된 루트파일시스템은 그림 5와 같이 기본 루트파일시스템은 기존 리눅스와 동일하게 압축된 상태로 플래시 메모리의 기존 주소에 저장하고, 확장 루트파일시스템은 플래시 메모리의 사용자 영역에 저장하는 구조를 가진다. 확장 루트파일시스템에 포함된 라이브러리 및 데이터는 RC 스크립터 단계에서 심볼릭(Symbolic) 링크를 구성하여 기존에 생성된 애플리케이션과 호환을 지원한다. 이 과정을 통해 RAM에서의 운영체제 영역의 크기를 줄일 수 있으며, 부팅 과정에서 압축 해제 및 RAM으로 복사되는 압축된 루트 파일시스템 크기를 최소화함으로써 부팅시간 감소 효과를 얻을 수 있다.

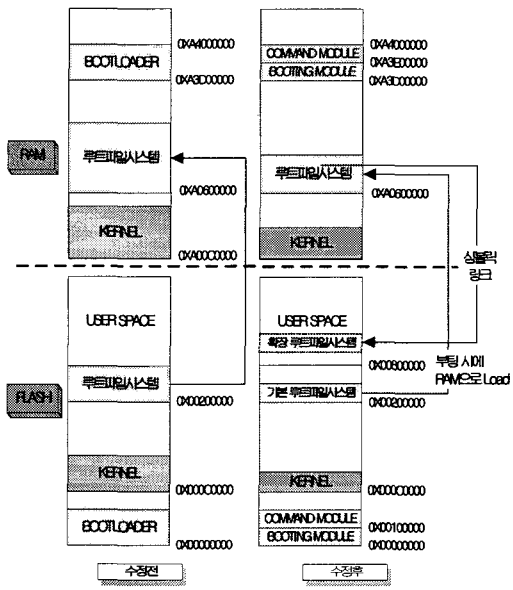


그림 5. 루트파일시스템의 메모리 맵 구조
Fig. 5. The architecture of root file system's memory map

표 3. 커널 configuration 최적화 예
Table 3. A sample of optimization for kernel configuration

상위 메뉴	하위 메뉴	최하위 메뉴	on / off
Kernel Hacking	Verbose user fault messages		off
	Kernel debugging		
	Include debugging information in kernel binary		
Sound	Sound support		
File Systems	DOS FAT fs support		
	Journalling Flash File System v2 (JFFS2)		
	/dev/pts file system for Unix98 PTYs		
	Network File Systems -> NFS file system support		
Input core support	Input core support		
ATA/ATAPI/MFM/RLL support	ATA/ATAPI/MFM/RLL support		
Plug and Play configuration	Plug and Play support		
General setup	Timer and CPU usage LEDs		
	Power Management support (experimental)		
	MMC devices drivers	Multi Media Card support	
	PCMCIA/CardBus support	PCMCIA/CardBus support	
	Support for hot-pluggable devices		
Code maturity level options	Pormpt for development and/or incomplete code/drivers		

Ⅳ. 성능 평가

본 논문에서는 테스트 임베디드 장비인 한백전자의 HBE-EMPOS II를 타켓 보드로 이용하여 제안한 기법의 성능을 실제로 측정하였다. 그리고 본 장에서는 임베디드 시스템 환경에서 리눅스 운영체제의 빠른 부팅을 지원하기 위해 제안한 부트로더 경량화기법, 커널 경량화 기법 및 루트파일시스템 경량화 기법을 각각 적용하여 기존의 테스트 장비에 탑재되어 있던 최적화되지 않은 임베디드 리눅스 운영체제와 비교 분석하였다.

4.1 실험 환경

실험 환경은 표 4에서 보는 바와 같이 타켓 보드와 호스트 PC로 이루어져 있다. 호스트 PC와 타켓 보드는 시리얼포트 및 JTAG 포트, 100MB 이더넷으로 연결되어 호스트 PC에서 생성된 부트로더, 커널, 루트파일시스템을 편리하게 타켓 보드에 탑재할 수 있도록 환경을 구축하였다. 또한 호스트 PC에는 Xscale용 크로스 컴파일인 arm-linux-gcc 환경이 구축되어 타켓 보드에 적합한 커널 및 부트로더를 컴파일하여 생성할 수 있도록 설정하였다. 커널 버전은 실험의 공정성을 위하여 HBE-EMPOS II에서 제공하는 임베디드 리눅스 운영체제와 동일한 리눅스 커널 버전 2.4.19를 대상으로 성능 평가 실험을 수행하였다.

표 4. 실험 환경
Table 4. A experimentation environment

구분	타켓 보드	호스트 PC
명칭	HBE-EMPOS II	IBM 호환 데스크 탑
프로세서	Xscale(PXA-255)	P4-1.8MHz
SDRAM	128MB	512MB
FLASH	32MB	-
부트로더	EMPOS-BOOT v1.0	LILO
Cross Complie	-	arm-linux-gcc
커널버전	2.4.19	

4.2. 성능 비교

본 논문에서 제안한 부트로더 경량화기법, 커널 경량화 기법 및 루트파일시스템 경량화 기법을 각각 적용하여 기존의 HBE-EMPOS II에서 제공하는 임베디드 리눅스 운영체제와 비교하였다. 그 결과 표 5에서 보는 바와 같이 각 구성요소 별로 최적화를 거친 후 부팅 시간이 전체 17초가 감소되는 성능의 향상이 있었다. 비교 대상인 HBE-EMPOS II용 임베디드 리눅스는 28초의 부팅시간이 소요된 반면에, 본 논문에서 제안한 기법을 타켓 보드인 HBE-EMPOS II에서 적용한 결과, 부트로더 과정에서 4초, 커널 단계에서 5초, 루트파일시스템 단계에서 8초로 전체 17초를 단축하는 성능 향상을 가져왔다. 또한 루트파일시스템이 RAM에서 차지하는 공간을 8.3MB에서 4.9MB로 줄임으로써 약 3.4MB의 RAM 공간을 추가로 확보할 수 있었다. 그러나 제안한 임베디드 리눅스의 전체 부팅 시간은 11초로서 파워 온과 동시에 정상적인 동작을 요구하는 임베디드 시스템에 탑재하기 위해서는 추가적인 성능 향상이 요구된다.

표 5. 실험 결과 비교
Table 5. A result of experimentation

	부트로더	커널	루트파일시스템
부팅시간 감소효과	4초 감소	5초 감소	8초 감소
크기	부팅최소모듈 (30KB ⇒ 3KB)	압축커널 (948KB ⇒ 696KB)	압축된 기본 루트파일시스템 (3172KB ⇒ 2180KB) RAM 상주 루트파일시스템 (8386KB ⇒ 4934KB)
부팅시간 변화	28초 (최적화 전) ▶ 11초 (최적화 후)		

Ⅴ. 결론 및 향후과제

본 논문에서는 리눅스를 탑재한 임베디드 시스템에서 빠른 부팅 메커니즘을 제안하여 적용함으로써 부팅 시간을 줄였다. 제안된 빠른 부팅의 핵심 메커니즘은 부팅에 필요한 최소의 모듈만 RAM으로 올리도록 리눅스의 구조를 변경하여 경량화한 것으로 RAM으로의 복사 및 압축 해제 시간을 줄여 부팅 시간을 줄이도록 하였다. 제안된 경량화 기법은 부트로더 경량화, 커널 경량화 그리고 루트파일시스템 경량화 단계로 나누어 크로스 컴파일러

환경이 지원되는 호스트 PC에서 타켓 보드인 HBE-EMPOS II에 적합하도록 구현하였다. 그리고 성능을 평가하기 위해 HBE-EMPOS II를 타켓 보드로 이용하여 제안한 기법의 성능을 실제로 측정하였다. 본 논문에서 구현된 결과는 기존의 HBE-EMPOS II에서 제공하는 임베디드 리눅스 운영체제와 비교하여 약 17초의 부팅 시간의 단축이 이루어지는 것을 확인하였다.

그러나 제안한 임베디드 리눅스의 전체 부팅 시간은 11초로서 파워 온과 동시에 정상적인 동작을 요구하는 임베디드 시스템에 탑재하기 위해서는 추가적인 성능향상이 요구된다. 즉, 보다 정밀한 커널 configuration 조정으로 커널 사이즈 최적화하는 연구 및 Source Level Kernel down sizeing 에 관한 연구, 루트 파일 시스템에서 동적 라이브러리의 효율적인 관리에 관한 연구는 이 문제를 해결할 수 있는 향후 과제가 될 것이다.

참고문헌

[1] Bill Roman, "Tips and Tricks for Implementing Software Execute-In-Place with Windows CE.NET", Datalight, 2003.

[2] 박지용, 이재수, 홍성수, 김동환, 장동은, "Shared Library and Execute-In-Place Support in MMU-less Embedded Systems", 2003 SoC Design Conference, pp. 722-727, 2003. 11.

[3] Chanik Park, Jaeyu Seo, Sunghwan Bae, Hyojun Kim, Shinhan Kim, Bumsoo Kim, "A low-cost memory architecture with NAND XIP for mobile embedded systems", Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system, pp. 138 - 143, 2003.

[4] 윤진혁, "NAND 플래시 메모리에서 XIP 기능의 지원", 서울대 대학원 석사학위논문, 2001.

[5] Jaesoo Lee, Jiyong Park, and Seongsoo Hong, "Memory Footprint Reduction with Quasi-Static Shared Libraries in MMU-less Embedded Systems", IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 24-33, San Jose, USA, April 2006.

[6] Holger Patecki, Peter Altenbernd, Michael Ditze, Reinhard Bernhardi-Grisson, "A Lightweight Linux Architecture for Resource-Limited Media Systems", Euromicro Conference on Real-Time Systems, 2003.

[7] Tim R. Bird, "Methods to Improve Bootup Time in Linux", Linux Symposium, 2004.

[8] David Selvakumar & Chester Rebeiro, "RTLinux on Memory Constraint Systems", Real Time Linux Workshop, 2004.

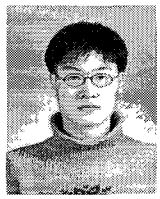
[9] Daniel P. Bovet, Marco Cesati, "Understanding The Linux Kernel", O'Reilly, 2003.

[10] 카림 야크무르, 김태석 역, "임베디드 리눅스 시스템 구축하기", 한빛미디어, 2004.

저자소개

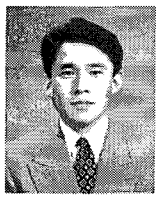
박 성 호(Seongho Park)

1996년 부산대학교 전자계산학과 졸업 (학사)
1998년 부산대학교 대학원 전자계산학과 졸업(이학석사)
2002년 부산대학교 대학원 전자계산학과 졸업(이학박사)
2002년 9월 ~ 현재 부산대학교 정보전산원 부교수
※관심분야 : VOD 시스템, 인터넷 캐쉉, 멀티미디어 통신, 비디오 트랜스코딩, 임베디드시스템



신 광 무(Kwangmu Shine)

2005년 부산대학교 전자전기정보컴퓨터 공학부 졸업(학사).
2005년~현재 부산대학교 컴퓨터공학과 석사과정.
※관심분야 : 임베디드 리눅스, 멀티미디어 시스템.



김 영 주(Yongju Kim)

1988년 부산대학교 계산통계학과(이학사)
1990년 부산대학교 계산통계학과(이학석사)
1990년 ~ 1995년 큐닉스컴퓨터 응용시스템연구소
1999년 부산대학교 전자계산학과 (이학박사)
2000년 ~ 현재 신라대학교 컴퓨터공학과 부교수
※관심분야 : 임베디드시스템, 멀티미디어처리, 영상 압축 및 통신