RFID 리더를 위한 경량 임베디드 리눅스 구현

신광무*○ 박성호** 정기동*
부산대학교 컴퓨터공학과*, 부산대학교 정보전산원**
{omega*○, kdchung* }@melon.cs.pusan.ac.kr shpark**@pusan.ac.kr

Light-weight Embedded Linux Implementation for RFID Reader

Kwangmu Shin*○ , Seongho Park** , Kidong Chung*
Dept. of Computer Engineering, Pusan National University*
Information Technology Center, Pusan National University**

요 약

본 논문은 임베디드 리눅스 (Embedded Linux)를 기반으로 하고 있다. 리눅스는 안정성, 유연성, 오픈 소스, 다양한 하드웨어 플랫폼 지원, 검증된 네트워크 등을 장점으로 임베디드 시스템 (Embedded System)의 운영체제로 많이 사용되고 있다. 하지만 기존의 리눅스 시스템은 중대형 시스템을 기반으로 운용되었기 때문에 자원 제약이 많이 따르는 임베디드 환경에서 적합하지 않다. 그리고 수십초가 걸리는 부팅시간도 중요한 문제점으로 작용한다. 본 논문은 임베디드 시스템인 RFID 리더 (Radio Frequency IDentification Reader)에서 경량화 (light-weighted) 과정을 거친 임베디드 리눅스를 운용할 수 있도록 하였다. RFID 리더의 임베디드 리눅스는 보다 적은 메모리를 사용하여 메모리 사용 효율성을 높였고 경량화 전의 시스템에 비교하여 상당한 부팅시간 감소 효과를 얻었다.

1. 서 론

임베디드 시스템이란, 시스템을 동작시키는 소프트웨어를 하드웨어에 내장하여 미리 정의된 특정 기능만을 수행한다. 즉, PDA, 휴대전화, 홈 네트워크 (Home Network), 디지털 TV, PMP (Portable Multimedia Player), 산업용 컨트롤러, 냉장고, 미사일 등의 제품 안에 간단한 컴퓨터가 들어 있고 그 컴퓨터 안에 임베디드 소프트웨어가 탑재되어 동작을 제어하는 것을 말한다.

본 논문은 임베디드 시스템의 한 부류인 RFID 리더에서 임베디드 리눅스를 운용할 수 있도록 제안하고 있다. RFID 기술은 유비쿼터스 (Ubiquitous) 사회를 만들어 가기 위한 핵심 기술로서 기초기반 기술 및 사회기반 기술의 정비가 진행되어 가고 있고, 현재 IT839 전략의 신성장 동력의 하나로 추진되고 있다. 기술적인 부분을 살펴보면, RFID 는 사물에 전자 태그를 부착하고, 각 사물의 정보를 수집, 가공함으로써 개체 간 정보 교환, 측위, 원격처리, 관리 등의 서비스를 제공하는 것이다. RFID 는 기본적으로 태그 (Tag)와 리더 (Reader), 그리고 미들웨어 (Middleware)로 구성되며, 이를 통해 장비나 사물에 정보를 저장하고 있는 초소형의 태그를 부착하고 이를 무선 기술을 이용한 리더로 읽어서 수집, 가공함으로써 소비자에게 다양한 서비스를 제공할 수 있게 된다.

실험에서 사용된 RFID 리더는 차세대물류 IT 기술연구사업단 산하 타 연구실에 제작한 433MHz 고정형 스마트 리더이다. RFID 리더도 일반 임베디드 시스템과 마찬가지로 자원 제약이 많이 따르기 때문에 기존 임베디드 리눅스의 경량화 과정이 필요하다. 그리고 RFID 리더가 물류 시스템에 적용될 경우에는 짧은 부팅시간도 아주 중요하다. RFID 리더의 예기치 못한 시스템 오류로 인해 재시작을 요구할 때 정상 가동을 위한 긴 지연 시간은 물류 비용을 증가 시킬 것이다.

임베디드 리눅스 경량화 과정은 크게 리눅스 커널 (Kernel)의 경량화, 루트파일시스템 (Root Filesystem)의 경량화, 부트로더 (Bootloader)의 경량화로 나눌 수 있다. 세 과정의 모든 경량화 과정은 거친 보드 RFID 리더 상에서의 실험 이전 단계인 XScale 프로세서를 탑재한 HBE-EMPOS II 보드이다. RFID 리더 상에는 현재 리눅스 커널의 경량화와 루트파일시스템의 경량화 과정만이 적용되어 있다. HBE-EMPOS II 보드에 경량화 과정이 모두 적용되어 있기 때문에 설계과정 및 실험결과에 그 내용을 같이 기술한다.

리눅스 커널 환경설정 최적화를 통해 리눅스 커널을 경량화 하였고, 부트로더는 개발용에 필요한 명령어 모드를 제거하여 경량화 하였다. 그리고 루트파일시스템은 바이너리 명령어 축소, 라이브러리 축소 등을 통하여 경량화 과정을 거쳤다.

경량화 과정만으로 부팅시간 절감 효과를 얻을 수 있지만, 기존 시스템과 차별적인 짧은 부팅시간을 얻기 위하여 부트로더 초기화 과정 변경 및 루트파일시스템의 RC 스크립트 변경을 과정을 더 추가하였다.

결과적으로 본 논문은 경량 임베디드 리눅스를 RFID 리더에 적용함으로써 보다 적은 자원으로도 시스템을 운용할 수 있게 되었고, 기존 임베디드 리눅스와 차별화된 빠른 부팅을 지원하게 되었다.

본 논문에서의 구성은 다음과 같다. 2장에서 관련 연구, 3장에서 임베디드 리눅스 경량화 설계, 4장에서 실험 결과, 그리고 마지막으로 5장에서 결론 및 향후 과제를 기술한다.

2. 관련 연구

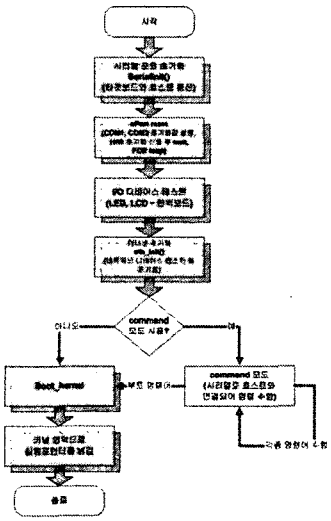
본 논문에서 임베디드 리눅스라고 언급하는 하는 것은 임베디드 리눅스 시스템을 의미한다. 즉, 임베디드 시스템에서 리눅스를 운용하기 위해 필요한 부트로더 및 루트파일시스템를 포함한다.

리눅스 커널을 경량화하는 방법으로 커널 환경설정 (configuration)을 최적화 (optimization) 하는 것을 들 수 있다 [1,2,11]. 즉, 커널 환경설정 부분을 임베디드 시스템의 목적에 맞게 최소 옵션만으로 설정하는 것이다. 본 방법을 사용하기 위해서는 우선 수십 종류의 각 세부 환경설정을 익히는 것이 중요하다. 그리고 포팅되는 임베디드 시스템의 목적을 파악하

"이 논문은 교육인적자원부 지방연구중심대학육성사업 (차세대물류IT기술연구사업단) 의 지원에 의하여 연구되었음."

여 포팅실험을 거쳐 가능한 작은 크기의 커널을 만드는 것이다.

부트로더는 수십 KB 에서 200KB 이하로 커널이나 루트파일시스템에 비해 크기가 작다. 하지만 자원 제한이 심한 임베디드 시스템 환경에서는 작은 모듈이라도 경량화 과정을 거쳐 제약 사항을 극복할 필요가 있다. 부트로더 경량화를 위한 최적화 단계 이전에 부트로더 구조 분석이 있어야 한다[4,5,9]. 본 논문의 실험 환경은 임베디드 보드인 "HBE-EMPOS II" 와 "RFID 리더" 를 기준으로 한다. HBE-EMPOS II 는 BLOB 부트로더를 사용하고, RFID 리더는 U-Boot 부트로더를 사용한다. BLOB 는 LART 하드웨어 프로젝트를 위한 부트로더로 소개되었다. 그리고 플래시 재프로그래밍을 위해 사용할 수 있고, JFFS2 MTD 파티션으로부터 커널을 직접 로드할 수 있지만 모니터 기능을 제공하지 않는 특징을 갖고 있다. BLOB 의 구조는 <그림 1>과 같다. U-Boot 는 PPC, ARM, MIPS, x86 등 아주 다양한 플랫폼에서 동작한다는 것을 장점으로 하고 있다.



<그림 1> HBE-EMPOS II 부트로더

루트파일시스템 경량화 과정 전에 선행되어 부분이 루트파일시스템 구조 분석이다[5]. 루트파일시스템의 최상위 레벨 구조는 아래 <표 1>과 같다.

<표 1> 루트 파일시스템 최상위 레벨 구조

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, FTP 등의 서비스를 위한 엔트리를 포함하기도 함
lib	C 라이브러리나 커널 모듈 등의 필수 라이브러리
mnt	일시적으로 마운트된 파일시스템에 대한 마운트 지점
opt	부가적인 소프트웨어 패키지
proc	커널과 프로세스 정보를 위한 가상 파일시스템
root	루트 사용자의 홈 디렉토리
sbin	필수적인 시스템 관리 바이너리
tmp	임시 파일
usr	X 서버를 포함해서 사용자에게 유용한 대부분의 응용 프로그램과 문서를 포함하는 서브 디렉토리
var	데몬과 유틸리티에 의해 저장되는 변수 데이터

임베디드 리눅스의 경량화와 밀접한 관련이 있는 빠른 부팅에 관한 연구는 다양한 테크닉을 내세워 시도되어 왔다 [3,6,7,8,10]. 예를 들면, XIP (eXecute-In-Place), Calibration 자연 감소, IDE 탐침 (Probing) 제거, "quiet" 옵션 사용, RC (Run Command) 스크립트 변경 등의 방법들이 있다. 그러나 이들 연구에서 제시하고 있는 방법들은 하드디스크를 사용하지 않는 임베디드 시스템에 적용할 수 없거나 현실적인 부팅시간 감소 효과가 미약하였다. 단, RC 스크립트 관련 방법은 적용하려는 임베디드 시스템에 맞게 수정하면서 상당한 부팅시간 감소 효과를 얻을 수 있는 방법이었다. 본 논문에서도 이 방법을 적용하여 부팅시간 절약 효과를 얻었다.

3. 임베디드 리눅스 경량화 설계

3.1. 리눅스 커널의 경량화

커널을 경량화 방법으로 채택한 것은 RFID 리더에 최적화된 커널 환경설정 구성이다. 아래 <표 2>는 듀얼 커널로 동작하는 RFID 리더의 리더/통신 모듈 중에서 리더 측의 커널 환경설정을 보여주고 있다. 리더 모듈에서는 네트워크 기능이 사용되지 않기 때문에 제거되었다. 커널 크기를 줄임으로써 램 복사 시간, 압축 해제 시간 등이 절약되어 부팅시간 감소 효과까지 얻을 수 있다.

<표 2> TimeSys 리눅스 커널 환경설정 최적화

상위 메뉴	하위 메뉴	OFF
TimeSys enhancements	Always acquire rwlocks for writing	
	Keep track of time spent in user and kernel mode	
	Kernel measurement tool	
Networking options	Netlink device emulation	
	Network packet filtering (replaces ipchains)	
	Socket Filtering	
	TCP/IP networking	
	QoS and/or fair queueing	
Network device support		OFF
Character devices	Unix98 PTY support	
File systems	Virtual memory file system support (former shm fs)	
	Ext3 journalling file system support (EXPERIMENTAL)	
	DOS FAT fs support	
	/dev/pts file system for Unix98 PTYs	
	Network File Systems	
	Journalling Flash File System v2 (JFFS2) support	
Kernel hacking		

3.2. 부트로더 경량화

부트로더 경량화 과정은 RFID 리더에 적용되기 이전인 HBE-EMPOS II 임베디드 실험 보드에서 적용되었다. <그림 1>에서 보는 바와 같이 각종 디바이스 초기화 및 테스트를 거친 후 오토 부팅 (Auto Booting) 여부를 결정하고 그 선택에 따라 커널 영역으로 실행 포인터를 넘기거나 명령어 모드를 실행한다. 명령어 모드는 부트로더, 커널, 파일시스템 등을 바꾸는 것처럼 개발 과정이 아니면 수행이 요구되지 않는다. 본 논문에서는 명령어 모드를 부트로더 이미지에서 제거하여 실제 부팅에 필요한 모듈만이 메모리에서 사용될 수 있도록 하였다. 리눅스 커널 경량화와 마찬가지로 부팅시간 감소 효과도 있었다.

3.3. 루트 파일시스템 경량화

루트파일시스템 경량화 과정에 사용하는 최적화 방법은 루트파일시스템을 부팅 시에 필요한 기본 영역과 부팅 후에 필요한 확장 영역으로 나누는 것이다. 본 최적화 방법은 RFID 리더에 적용되기 이전인 HBE-EMPOS II 임베디드 실험 보드에서 적용되었다. 기본 영역은 압축 램디스크에 포함시켜 플래시 메모리의 기존 주소에 올린다. 이 과정에서 램디스크 크기도 시스템 운용에 필요한 정도로 줄였다. 확장 영역에는 플래시 메모리의 유저 영역에 올리도록 하였다. 부팅 과정에서 기본 영역은 램으로 복사되지만 확장 영역은 플래시 메모리에 그대로 남

게 된다. 부팅 후의 파일시스템 간 연결은 심볼릭 링크(Symbolic Link)로 유지하였다. 그리고 개발 과정 상에서만 필요한 기타 명령어 및 라이브러리를 제거하여 루트파일시스템의 크기를 줄이도록 하였다. 본 방법은 부팅 과정에서 기본 영역만이 관여하기 때문에 확장 영역만큼의 램 복사 시간 및 압축 해제 시간을 절약할 수 있다. 즉, 부팅 시간을 단축시키는 효과를 가져올 수 있었다.

3.4. 빠른 부팅 방법

임베디드 리눅스의 각 모듈 별 경량화 과정은 부팅 시간 감소에도 영향을 미친다. 본 논문에서는 임베디드 시스템에서 빠른 부팅을 구현하기 위하여 경량화와 병행하여 몇 가지 방법을 같이 적용하였다. 부트로더에서 초기화 과정 최적화, 루트파일시스템의 RC 스크립트 최적화이 그것이다. RFID 리더에서는 RC 스크립트 최적화 과정만이 적용되었다. RC 스크립트 최적화란, 임베디드 시스템의 목적에 부합되지 않게 수행되는 여러 서비스를 실험을 통하여 제거시킴으로써 부팅 시간을 단축시키는 것을 말한다.

4. 실험 결과

본 실험에서 사용된 시스템은 HBE-EMPOS II 임베디드 보드와 Virtex-II Pro FF1152 를 기본으로 하는 RFID 리더이다. 각 시스템의 간단한 특성은 <표 3>에서 보는 바와 같다. RFID 리더는 듀얼 코어 (Dual Core)를 갖추고 있고 각 코어가 리더/통신 모듈을 개별적으로 관리한다. 다음 결과에서의 실험 환경은 리더 모듈을 기준으로 하고 있다.

<표 3> 실험 환경

	HBE-EMPOS II	RFID 리더 (Virtex-II Pro FF1152)
램	128MB	64MB
플래시 메모리	32MB	16MB
코어	ARM	PPC405 듀얼
부트로더	EMPOS-Boot	U-Boot
커널	2.4.19	2.4.18 (TimeSys Linux)

임베디드 리눅스의 경량화 결과는 <표 4>에서 보는 바와 같다. HBE-EMPOS II 와 RFID 리더 모두, 커널에서 300KB 이상의 경량화 결과를 얻을 수 있었다. 차후 실험 진행에 따라 RFID 리더에서도 HBE-EMPOS II 에서 적용한 것과 같이 나머지 경량화 방식을 적용할 수 있을 것이다.

그리고 <표 5>에서 보는 바와 같이 경량화 과정과 부팅 시간 개선 방법을 적용하여 상당한 정도의 부팅 시간 절약 효과를 얻을 수 있었다. HBE-EMPOS II 경우에는 거의 최적화 과정 전과 비교하여 거의 반 정도의 부팅 시간을 보여 주고 있다. 부팅 시간은 시리얼 라인으로 보여지는 부팅 메시지에서 특정 문자를 인식하여 측정하였다.

<표 4> 임베디드 리눅스 모듈 별 크기

	HBE-EMPOS II		RFID 리더 (Virtex-II Pro FF1152)	
커널	941KB	604KB	901KB	587KB
부트로더	28KB	5KB		
루트파일시스템 (압축 램디스크)	3169KB	2182KB		

<표 5> 부팅 시간

	최적화 전	최적화 후
HBE-EMPOS II	27초	14초
RFID 리더 (Virtex-II Pro FF1152)	20초	14초

5. 결론 및 향후과제

임베디드 시스템의 성능이 많이 향상되고 있지만 아직 자원의 제약은 크다. 이에 부합하여 본 논문은 RFID 리더에서 경량화된 임베디드 리눅스를 동작시킬 수 있도록 초점을 맞추었다. 커널 경량화 과정 실험에서 RFID 리더는 경량화 이전과 같이 정상적으로 동작하였다. 나머지 모듈의 경량화 과정도 HBE-EMPOS II 에서와 같이 수행된다면 RFID 리더 동작에는 문제가 없을 것이다. 그리고 임베디드 시스템에서 자원을 보다 효율적으로 사용할 수 있는 경량 임베디드 리눅스 구현이 될 것이다.

그리고 임베디드 리눅스 경량화와 몇 가지 테크닉을 이용하여 RFID 리더에서 중요한 문제인 빠른 부팅을 구현할 수 있었다. 앞으로 RFID 리더에 경량화 과정을 더 진행해 나간다면 더 나은 부팅 시간 단축 효과를 얻을 수 있을 것이다.

[참고 문헌]

[1] Holger Patecki, Peter Altenbernd, Michael Ditze, Reinhard Bernhardi-Grisson, "A Lightweight Linux Architecture for Resource-Limited Media Systems"

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

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

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

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

[6] 박찬익, "A low-cost memory architecture with NAND XIP for mobile embedded systems", ACM, 2003

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

[8] http://tree.celinuxforum.org/

[9] http://kelp.or.kr/

[10] http://www-128.ibm.com/developerworks/linux/library/l-boot.html

[11] http://kldp.org/KoreanDoc/html/Kernel-KLDP/index.html