휴대폰을 위한 임베디드 리눅스 경량화 기법*

임윤재° 남영진† 김성률‡ 서대화‡
경북대학교 전자공학과, †대구대학교 컴퓨터·IT공학부
‡임베디드소프트웨어 협동연구센터
launius@ee.knu.ac.kr°, yjnam@daegu.ac.kr, srkim@escrc.re.kr, dwseo@ee.knu.ac.kr

Techniques for Light-weighted Embedded Linux for Mobile Phones

Yunjae Lim°, Youngjin Nam†, Sungryul Kim‡, Daewha Seo‡
Department of Electronics Engineering, Kyungpook National University
†School of Computer & Information Technology, Daegu University
‡Embedded Software Cooperative Research Center

요 약

임베디드 리눅스 운영체제는 전형적인 임베디드 시스템 중의 하나인 휴대폰 환경에도 적용될 수 있다. 실제로 최근 휴대폰 제조사들이 휴대폰에 리눅스를 채택하기 시작하면서 리눅스 기반의 운영체제를 탑재한 휴대폰이 다수 개발되어 휴대폰 시장에서 꾸준한 성장 추세를 이어가고 있다. 임베디드 리눅스를 휴대폰 환경에 적용하기 위해서는 몇 가지 기술적인 요구사항들이 고려되어야 하며, 대표적으로 리눅스 시스템 크기에 대한 경량화가 필요하다. 본 논문에서는 임베디드 리눅스를 휴대폰 환경에 적용하기 위한 구체적인 경량화 기법들을 제시하고 각 기법을 휴대폰 리눅스 플랫폼 환경에 적용하여 커널과 루트 파일 시스템 각각의 ROM, RAM 사용량 측면에서 그 효과를 검증해 본다.

1. 서 론

임베디드 리눅스는 우수한 안정성과 폭넓은 개발 용이성을 바탕으로 이미 산업용, 사무용, 가정용 등 많은 임베디드 시스템 분야에서 널리 사용되고 있다. 초기의 임베디드 운영체제 시장은 운영체제의 크기가 작은 RTOS(Real-Time Operating Systems) 위주로 형성되어 왔다. RTOS의 경우, 적은 용량의 메모리에도 탑재가 가능하여 생산 비용을 줄일 수 있었기 때문이다. 최근에는 기술의 발전으로 보다 적은 비용으로도 대용량의 고집적 IC 메모리와 고성능 마이크로프로세서를 사용할 수 있게 되었고, 임베디드 운영체제 시장도 전통적인 RTOS 위주에서 리눅스와 같은 표준 운영체제의 경량 버전으로 확대되고 있다[1].

이와 같은 임베디드 리눅스는 전형적인 임베디드 시스템 중의 하나인 휴대폰 환경에도 적용될 수 있다. 실제로 최근 휴대폰 제조사들이 휴대폰에 리눅스를 채택하기 시작하면서 리눅스 기반의 운영체제를 탑재한 휴대폰이 다수 개발되어 휴대폰 시장에서 꾸준한 성장 추세를 이어가고 있다. 휴대폰 운영체제 시장은 현재 심비안이 장악하고 있고 그 뒤를 마이크로소프트가 추격하고 있는 양상이다. 임베디드 리눅스를 사용한 휴대폰은 리눅스를 휴대폰 환경에 맞게 재조정하는 작업에 상당 기간이 소요되어 보급 속도가 기대보다는 느리지만 지속적인 성장추세를 보여주고 있다. IDC 조사에 따르면 2006년까지는 리눅스가 고성능 스마트폰용 소프트웨어 시장의 약 4.2%를 차지할 것으로 추정되고 있다.

임베디드 리눅스가 실제 여러 분야의 제품에 탑재되어 사용되고 있으나, 일부 제품에서는 몇 가지 제약사항들로 인하여 임베디드 리눅스를 사용하기에 어려움이 있다. 그 대표적인 제약사항이 리눅스의 크기이다. 셋톱박스와 휴대폰, 데스크탑 PC의 하드웨어 환경을 비교한 표 1에서 볼 수 있듯이, 휴대폰과 같은 소형 전자제품은 하드웨어 환경이 PC와는 상당히 다르며 상대적으로 부피가 큰 리눅스 시스템을 휴대폰 상에 그대로 적용하기에는 다소 무리가 따른다.

임베디드 리눅스를 휴대폰 환경에 적용하기 위해서는 몇 가지 기술적인 요구사항들이 고려되어야 하며, 대표적으로 리눅스 시스템 크기에 대한 경량화가 필요하다. 따라서, 본 논문에서는 임베디드 리눅스를 휴대폰 환경에 적용하기 위한 기술적인 요구사항들을 살펴보고, 리눅스 시스템의 경량화를 위한 구체적인 기법을 제시한다. 또한, 경량화 기법들을 휴대폰 리눅스 플랫폼 환경에 적용하여 커널과 파일 시스템 각각의 ROM, RAM 사용량 측면에서 그 효과를 검증해 본다.

표 1. 소형 전자제품과 데스크톱 PC의 하드웨어 환경 비교[2]

	Set Top Box(STB)	Cellular Phone	Desktop PC
CPU(frequency)	MIPS(30MHz)	ARM(50MHz)	Pentium(2 200MHz)
RAM size	32Mbyte	8Mbyte	512Mbyte
Flash/ROM	16Mbyte	4Mbyte	-
Storage	HDD 15Gbyte	none	HDD 80Gbyte

본 논문의 2장에서는 임베디드 리눅스를 휴대폰에 적용하기 위해 필요한 요구사항들을 살펴본다. 3장에서는 리눅스 시스템의 경량화를 위한 구체적인 방법들을 제시하고 4장에서 커널과 루트 파일 시스템 각각의 ROM, RAM 사용량 측면에서 적용 결과를 검증한 후, 5장에서 결론을 맺는다.

2. 휴대폰을 위한 임베디드 리눅스 요구사항

현재 모바일 디바이스의 대표적 제품이라고 할 수 있는 휴대폰은 부품 기술과 플랫폼의 발달로 컴퓨터와 거의 동등한 성능을 갖춘 기기로 발전하고 있으며, 이에 따라 프로그램, 게임 등과 같은 컨텐츠 역시 복잡해지고 있다. 휴대폰이 기본적인 통신 기능과 그래픽 인터페이스 뿐만 아니라 한층 더 업그레이드 된 멀티미디어, 게임 등의 기능을 포함하게 됨에 따라 이를 뒷받침할 수 있는 강력한 운영체제가 필수적으로 요구된다. 따라서, 휴대폰 운영체제로 리눅스와 같은 표준 운영체제의 경량 버전이 적합하게 사용될 수 있으며, 이를 위해서는 임베디드 환경의 리눅스를 휴대폰 환경에 맞게 재조정하는 작업이 필요하다. 일반적으로 파워 소비, 메모리 사용량, 부팅 시간의 단축, 실시간 성능, 전원 종료와 재부팅 시의 안정성, 보안 등과 같은 사항들이 고려되어야 한다.

리눅스를 사용한 휴대폰 상에서 다양한 어플리케이션을 구현한 소프트웨어 구조는 그림 1과 같다. 휴대폰에 필요한 기본기능 뿐만 아니라, 추가적으로 멀티미디어 기능과 같은 업그레이드 기능

* 본 연구는 경북대 임베디드S/W협동연구센터 협동연구개발과제와 정보통신부 및 정보통신연구진흥원의 대학IT연구센터지원사업의 연구결과로 수행되었음 (IITA-2005-C1090-0501-0018)

이 구현되면서 휴대폰의 소프트웨어 구조는 한층 더 복잡해진다. 전화 통신, 멀티미디어, 시스템과 관련된 주변장치 드라이버는 응용 프로그램들 사이에서 공유되며 휴대폰 운영체제의 최상위 계층에 어플리케이션 프로그램들이 위치한다.

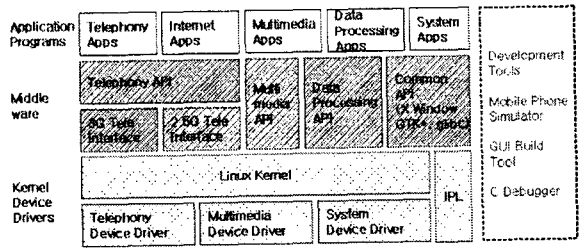

그림 1. 휴대폰 소프트웨어 구조[3]

실제로 휴대폰에서 사용되는 리눅스의 데이터 비율을 표 2에 나타내었다. 각 부분의 정확한 크기는 공개되어 있지 않아 알 수 없지만, 아래 표로부터 간접적으로 상대적인 크기를 추측해 볼 수 있다. 현재 사용되고 있는 리눅스 기반의 휴대폰에서 순수 휴대폰 기능을 위한 프로그램 영역(text 또는 code)의 크기 비율이 'mobile' 항목에 나타나있다. 표는 개발 초기버전에 대한 데이터를 나타내고 있어 어플리케이션 계층이 보다 커질 수 있음을 고려해야 한다.

표 2. 리눅스 기반 휴대폰의 데이터 비율[2]

Application	Mobile	10%	17%
	Internet	5%	
	Multimedia	2%	
Middleware	Mobile	26%	47%
	X/XIM/Fonts/Toolkit	17%	
	Multimedia	2%	
	Internet	1%	
Kernel/Userland	Command/glibc	28%	37%
	Driver module	4%	
	Kernel	4%	

표에서 확인할 수 있듯이 커널과 디바이스 드라이버, 커맨드/glibc를 포함하는 전체 운영체제 부분은 미들웨어 계층 전체에 필적하는 비율을 차지하고 있다. 리눅스 커널은 휴대폰 프로그램에서 차지하는 비율이 크게 높지 않으며, 전체 영역에서 약 5% 정도의 크기를 가진다.

3. 휴대폰을 위한 임베디드 리눅스 경량화 방법

본 절에서는 커널과 루트 파일 시스템을 포함한 리눅스 시스템의 크기를 줄이기 위한 설정들과 기법들에 대해 그 크기를 측정하고 비교하는 방법을 정의한다. 또한, 리눅스 시스템의 커널, 디바이스 드라이버, 커맨드/glibc 부분을 경량화 할 수 있는 구체적인 방법들을 제시하고 각 방법을 휴대폰 임베디드 리눅스 플랫폼 환경에 적용하여 성능을 분석해 보고자 한다.

3.1 측정 방법

임베디드 리눅스 경량화 기법을 정의하고 시스템 크기를 측정할 때, 몇 가지 사항을 고려해야 한다. 우선 경량화 기법의 대상이 ROM인지 RAM인지를 고려해야하고, 커널을 대상으로 하는지 파일시스템을 대상으로 하는지, 혹은 둘 다인지를 명확히 할 필요가 있다. 본 논문에서는 Kernel ROM size, Kernel RAM size, FS ROM size 측면에서 경량화 효과를 검증하며 size와 nm 툴을 사용하여 크기를 측정하고 비교한다. size 툴을 사용하여 프로그램의 obj 파일에서 text 영역, data 영역, bss 영역의 크기를 확인할 수 있으며, nm 툴로 obj 파일에서 각각의 함수들이 차지하는 크기를 확인할 수 있다.

3.2 경량화 방법

■ LinuxTiny[4,5] 프로젝트는 리눅스가 소형 시스템에서도 동작할 수 있도록 커널 2.6을 기반으로 커널의 메모리와 디스크 사용량을 경량화하는 패치셋을 제공한다. 패치는 i386 기반의 리눅스 커널 버전 2.6에 적용될 수 있도록 구현되었으며, 임베디드 시스템 개발자나 소형 디바이스, 레거시 시스템 사용자들을 주 대상으로 한다. 여기에서는 LinuxTiny 패치를 ARM 기반의 휴대폰 임베디드 리눅스 플랫폼 환경에 적용시켜 커널의 ROM과 RAM 크기를 경량화한다. LinuxTiny 패치셋 중에서 경량화 효과가 큰 아래 패치들이 적용되었다.

• kill-printk.patch : 커널 이미지에서의 메시지 출력을 제거한다. asmlinkage 타입의 printk 함수를 내부에 return 부분만을 가지는 inline printk 함수로 대체하고, kernel/printk.c 파일의 log_buf 관련 변수들과 console, log 관련 함수들을 제거하여 커널의 ROM과 RAM 사용량을 줄이고 실행속도를 높인다.
• no-aio.patch : 쓰레드 어플리케이션에서 성능을 높이기 위해 사용되는 POSIX 비동기 IO 함수들을 제거하여 커널크기를 줄인다. fs/aio.c 파일의 aio_complete 함수, exit_aio 함수 등과 kernel/sysctl.c 파일 fs_table 변수의 aio 관련 부분이 제거된다.
• no-attr.patch : 파일 시스템 확장 애트리뷰트 시스템 콜에 관련된 함수를 제거한다. fs/Makefile에서 xattr.o 오브젝트 파일 관련 부분이 제거된다.
• posix-timers.patch : 주로 실시간 어플리케이션에서 사용되는 POSIX 타이머 관련 API 함수를 제거한다. kernel/posix-timers.c 파일에서 posix_timers_id, posix_clocks, init_posix_timers와 같은 POSIX 타이머 관련 선언과 변수, 함수들을 모두 제거하여 커널의 크기를 줄인다.

■ 임베디드 환경에서는 리눅스 커널을 zImage와 같은 압축된 형태로 타겟 시스템의 플래시 메모리에 저장하여 커널의 ROM 크기를 경량화 할 수 있다. 리눅스 커널 소스를 컴파일하여 ELF32 executable 포맷의 압축되지 않은 커널 이미지 vmlinux를 생성한다. vmlinux는 objcopy 명령에 의해 바이너리 포맷으로 변환되고 gzip으로 압축되어 ELF32 relocatable 파일인 piggy.o를 생성한다. 여기에 압축을 해제하기 위한 함수들과 32bit 스타트업 코드를 묶어 ELF32 executable 포맷의 vmlinux 파일이 만들어지고, 최종적으로 objcopy 명령으로 바이너리 커널 이미지인 zImage 파일로 변환된다. 플래시 메모리에 저장된 압축 커널 이미지는 RAM 영역으로 복사, 압축 해제되어 실행된다.

■ gcc 컴파일러의 최적화 옵션을 사용하여 실행 파일의 크기, 컴파일 시간, 실행 속도와 같은 조건을 고려하여 선택적으로 프로그램을 빌드한다. gcc 컴파일러는 표 3과 같은 여러 단계의 최적화 옵션을 제공한다. 코드 크기를 최소화할 수 있는 최적화 옵션을 사용하여 커널 이미지의 ROM 크기를 경량화한다.

표 3. gcc 컴파일러 최적화 옵션

-O	-O1 옵션과 동일
-O1	컴파일 시간은 고려하지 않고 코드 크기를 줄이고 실행 속도를 향상시는 것을 목표로 최적화를 수행한다.
-O2	컴파일러에서 지원되는 거의 모든 최적화를 수행한다. 루프 전개, 함수의 인라인 전개, 레지스터 리네이밍 부분은 제외된다. 컴파일 시간이 다소 오래 걸리고 생성 코드의 크기가 증가한다.
-O3	-O2 옵션의 모든 최적화를 수행하고 루프 전개, 함수의 인라인 전개, 레지스터 리네이밍도 수행한다.
-O0	최적화를 수행하지 않는다.
-Os	코드 크기를 최적화한다. -O2 옵션에서 수행되는 최적화 중 코드 크기를 증가시키지 않는 최적화를 모두 수행하며, 코드 크기를 줄이기 위해 설계된 추가적인 최적화 옵션을 수행한다.

■ Initrd(INITial RamDisk)는 루트 파일 시스템을 램디스크 형태로 사용하는 커널의 특징이다. Initrd와 gzip압축을 사용하여 기존의 루트 파일 시스템이 저장되는 ROM 영역을 줄일 수 있다. 루트 파일 시스템 이미지가 압축된 상태로 ROM/Flash에 저장되면, 부트로더(혹은, 커널)가 압축된 이미지를 RAM으로 복사하여 압축을 풀고 파일 시스템으로 마운트하여 실행한다.

■ 루트 파일 시스템의 ROM 크기를 줄이는 방법으로서 읽기 전용 파일 시스템인 Cramfs을 이용한다. Cramfs은 압축된 파일 시스템이므로 ROM의 사용량을 줄일 수 있으나, 실행 시에 ROM에서 RAM으로 로딩 시간이 오래 걸리는 단점이 있다.

■ 루트 파일 시스템의 ROM/Flash 크기를 줄이는 또 하나의 방법으로 jffs2를 사용할 수 있다. jffs2는 플래시 메모리에서 사용되는 읽고 쓰기가 모두 가능한 압축 파일 시스템으로서 Memory Technology Device(MTD)를 사용하여 플래시 메모리의 수명을 절약할 수 있는 장점도 가진다.

■ 루트 파일 시스템의 실행 가능한 바이너리 파일에서 디버깅 정보를 담은 심볼릭 부분을 제거하여 ROM 크기를 경량화한다. strip 명령을 사용하여 심볼릭 참조 정보를 제거하면 실행 파일의 크기는 작아지지만 디버깅이 불가능해진다는 단점이 있다.

4. 경량화 방법에 대한 성능 평가

앞서 제시한 여러 가지 리눅스 시스템 경량화 방법을 CDMA 무선 통신 모듈이 부착된 T-Box Xt 임베디드 보드와 ARM 패치와 보드 패치가 적용된 리눅스 커널 버전 2.6.8.1을 사용하여 각각의 성능을 분석한다. 성능평가에 사용된 T-Box Xt는 Intel PXA255 Xscale 프로세서, Intel StrataFlash E28F128 NOR 플래시 메모리 32M(16MB×2), 삼성 K4S561632 SDRAM 64M(32MB×2), 3.5인치 해상도 240×320 TFT LCD의 하드웨어 환경을 가지고 있다.

■ i386 기반의 LinuxTiny 패치를 ARM 기반의 T-Box Xt 임베디드 리눅스 커널에 적용한 결과를 표 4에 나타내었다. 리눅스 커널에 LinuxTiny 패치를 적용한 후, 압축되지 않은 커널 이미지 파일 vmlinux를 생성시켜 패치 전과 후의 크기를 비교한다.

표 4. LinuxTiny 패치 적용 결과

Name of the patch	text	data	bss	total (dec)	ratio (total)	reduced size (total)
Stock kernel(2.6.8.1)	2541388	386480	113560	3041428		
kill-printk.patch	2293580	379136	96152	2768868	91.0	272560
no-alo.patch	2533884	385792	113528	3033204	99.7	8224
no-attr.patch	2539020	386400	113560	3038980	99.9	2448
posix-timers.patch	2533828	385488	113336	3032652	99.7	8776
All patch	2276396	377296	95928	2749620	90.4	291808

측정결과는 kill-printk.patch 파일을 적용하였을 때, 가장 큰 크기 감소를 보여준다. 4개의 패치를 모두 적용하였을 때는 전체적으로 약 10%의 커널 경량화 효과를 확인할 수 있다. 패치가 적용된 기능들은 일반적으로 자주 사용되지 않아 경량화 가능한 기능들이며, 각각의 패치들은 Kconfig 파일에 등록하고 소스코드에 ifdef 문으로 추가하여 필요시에는 menuconfig에서 선택적으로 적용할 수 있도록 구현하였다.

■ T-Box Xt 환경에서 리눅스 커널 2.6.8.1을 사용하여 zImage를 생성한다. 압축되지 않은 커널 이미지 vmlinux 파일을 압축된 커널 이미지 형태인 zImage로 변환시켰을 때의 크기 변화를 표 5에 나타내었다. 압축되지 않은 vmlinux는 약 3.75MByte 크기를 가지며, 바이너리 포맷의 압축 파일인 piggy.o로 변환하였을 때, 약 1.35MByte 크기를 보인다. 여기에 압축 해제와 스타트업 코드가 추가하여 생성된 zImage는 1.36MByte 정도의 크기를 가진다. 초기의 압축되지 않은 커널 이미지에 비해 최종적으로 약 63.8%의 경량화 된 크기를 확인할 수 있다.

표 5. 커널 이미지 압축 결과

Kernel Image File	size(byte)
$(TOPDIR)/vmlinux	3753068
$(TOPDIR)/arch/arm/boot/compressed/piggy.o	1348390
$(TOPDIR)/arch/arm/boot/compressed/vmlinux	1401803
$(TOPDIR)/arch/arm/boot/zImage	1360180

■ 표 6에 여러 가지 gcc 컴파일러의 최적화 옵션을 사용하여 생성된 커널 이미지 크기를 나타내었다. 최적화 옵션 -O, -O1을 사용하였을 때, 가장 작은 커널 이미지 크기를 확인할 수 있다. 이것은 -O1 옵션이 중점적으로 크기를 줄이는 기능을 수행하기 때문이며, 다른 최적화 기능들과 컴파일 시간, 수행속도 등 여러 가지 면을 고려할 때는 -Os 최적화 옵션이 성능에 비해 상당히 작은 크기를 보여준다.

■ T-Box Xt의 루트 파일 시스템에 Initrd와 gzip 압축 형태, Cramfs, jffs2 파일 시스템을 적용한 결과를 표 7에 나타내었다. 세 가지 방법에서 모두 비슷한 압축 성능을 확인할 수 있으며 압축하지 않았을 때에 비해서 약 60% 정도 크기가 줄어든다. 실험 결과는 Cramfs 이미지 크기가 가장 작으며, jffs2가 약간 더 큰 크기를 나타내고 있다. 그러나, 실제 적용에 있어서는 Cramfs은 읽기 전용 파일 시스템이며 실행 시에 ROM에서 RAM으로 로딩 시간이 오래 걸리는 반면, jffs2는 읽고 쓰기가 모두 가능하며 MTD를 사용하여 플래시 메모리의 수명을 절약할 수 있다는 특성을 고려할 필요가 있다.

표 6. gcc 옵션 적용 결과

gcc Option	size(byte)
-O, -O1	1325216
-O2	1439420
-O3	1685516
-Os	1360180

표 7. 루트 파일 시스템 압축 결과

File System	size(byte)
Initrd, gzip Image (Not stripped)	2691663
Jffs2 Image (Not stripped)	2737096
Cramfs Image (Not stripped)	2535424
Uncompressed & Not stripped Root File System	6217728
Uncompressed & Stripped Root File System	6131712

5. 결론 및 향후연구

본 논문에서는 휴대폰에 임베디드 리눅스를 적용하기 위해서 커널, 디바이스 드라이버, 커맨드/glibc 부분을 경량화할 수 있는 구체적인 방법들을 제시하고 각 방법을 휴대폰 임베디드 리눅스 플랫폼 환경에 적용하여 성능을 분석하였다. 향후연구로 휴대폰 어플리케이션에서 사용할 수 있는 공간을 늘리기 위해서는 운영 시스템 부분만이 아니라 상위 미들웨어 부분에 대해서도 경량화에 대한 추가적인 작업을 수행할 계획이다.

참고문헌

[1] S. Santo, "Embedded battle royal," IEEE Spectrum. Vol.38, pp.36-41, Dec. 2001.
[2] SystemSizeSpec_R2, http://tree.celinuxforum.org/pubwiki/ moin.cgi/SystemSizeSpec_5fR2.
[3] Y. Nakamoto, "Technical challenges toward the next generation mobile phone linux," Proc. of Computer and Information Technology, Sep. 2004.
[4] LinuxTiny, http://www.selenic.com/linux-tiny/.
[5] M. Mackall, "Linux-tiny and directions for small systems," Proc. of the Linux Symposium, Jul. 2004.