임베디드 리눅스 기반 단말기의 빠른 부팅 개선 방법

이광로*, 배병민*, 박호준**

*㈜스마트큐, **한국전자통신연구원

Fast booting solution with embedded linux-based on the smart devices

GowangLo Lee*, ByeongMin Bae*, HoJun Park**

*SmartQ Technologies, **ETRI

Email : lgllee@smartqtech.com, baemin@smartqtech.com, ho.joon.park@etri.re.kr

요 약

본 논문에서는 임베디드 리눅스 기반 단말기의 빠른 부팅 개선을 위해 부팅 과정을 부트로더, 커널, 파일 시스템, 초기화 스크립트, 공유 라이브러리, 응용 프로그램 등 6가지 단계로 나누었다. 빠른 부팅 개선을 위해 전원인가 시 최초로 실행되는 부트로더 단계와 초기화 스크립트 단계에 적용했다. 부트로더 단계에서 입력 대기 시간 제거, 불필요한 초기화 루틴제거, 커널 이미지 비압축 로드, 최적화된 복사 루틴 사용 등을 적용하여 부팅 개선을 했다. 또한 초기화 스크립트 단계에서 이진화 기반 스크립트 대체 기술 사용, init 프로세스 경량화 등을 적용하여 부팅 개선을 했다.

ABSTRACT

In this paper, we propose a fast booting solution with embedded linux-based smart devices. We have divided the fast boot process into six steps, such as boot loader, kernel, file system, the init-scripts, shared libraries, and applications for an embedded linux-based boot process to improve the fast booting. Improvements for the fast boot are made in the boot loader phase, which is the first phase at power-up, and the init-script that runs the boot loader phase. To improve the fast booting, standby time from the boot loader and unnecessary initialization routine have been removed, and uncompressed kernel image loading as well as optimized copy routine have been applied. Further, a technology that replaces binary scripts in init-script phase and light-weight init process have been utilized to improve the boot.

키워드

Keyword: Embedded system, fast booting, boot loader, kernel, file system, NAND flash

Ⅰ. 서 론

임베디드 리눅스 시스템을 사용하여 휴대전화, 개인 정보 단말기(PDA), 미디어 플레이어, 셋톱 박스, 그 밖의 가전 기기와 같은 다양한 제품이 제작되고 있다. 임베디드 리눅스를 채택한 제품들에 대해 빠른 부팅을 채택한 제품이 선호도와 구매력을 더 가짐으로써 빠른 부팅에 대한 관심도가 점차 높아지고 있다[1][2][4].

리눅스의 부팅 과정은 각 단계의 독립성, 일반적인 데스크탑 형태의 컴퓨터 시스템에서 필요한 유연성 등을 포함하고 있어야 하기 때문에 매우 복잡하다. 임베디드 리눅스가 탑재된 임베디드 시스템은 데스크탑이나 서버 형태의 리눅스와는 달리 그 용도가 한정적인 것이 특징이다[3]. 따라서 사용자가 개입할 필요가 없어지므로 임베디드 리눅스에서의 부팅과정에서는 쉘스크립트는 선택적이다. 위와 같이 임베디드 리눅스 상의 부팅 과정은 데스크탑 형이나 서버 형의 리눅스의 부팅 과정을 축약하여 불필요한 작업을 하지 않거나 시간을 줄이는 형태를 취하고 있다.

본 논문에서는 임베디드 리눅스 기반 단말기의 부팅 개선을 위해 부팅 과정을 부트로더, 커널, 파일 시스템, 초기화 스크립트, 공유 라이브러리, 응용 프로그램 등 6가지 단계로 나누었으며, 빠른 부팅 개선을 위해 전원인가 시 최초로 실행

되는 부트로더 단계와 초기화 스크립트 단계에 적용했다.

Ⅱ. 부팅 개선방법 I: 부트로더 단계

부트로더는 최초 전원인가 시 실행되는 임시성을 지닌 프로그램으로 본 논문에서는 U-boot를 사용한다. U-boot는 대부분의 임베디드 시스템에서 작동되는 코드를 위해 pcc, ARM, MIPS를 지원하게 끔 제작된 소프트웨어이다. 부트로더에서 사용되는 기술은 입력 대기 시간 제거, 불필요한 초기화 루틴제거, 커널 이미지 비압축로드, 최적화된 복사 루틴 사용 등 4가지이다.

2.1 입력 대기 시간 제거

U-boot는 자체 수정을 위해 Command Mode와 일상적인 Boot Mode가 존재한다. 이중 커맨드 모드로 가기 위해 사용자 입력대기 시간을 두게 되는 것이 보통이다. 만일 지정된 시간 동안 디버그 포트 등을 통해 사용자가 아무런 입력을 하지 않으면 일상의 부트 모드로 가고 입력을 하면 U-boot의 프롬프트가 나와 사용자의 명령을 기다리게 된다. 그러나 이 시간은 개발이 완료가 되어 제품이 양산 시점에 이르면 더 이상 필요 없는 것이므로 삭제한다.

적용 및 기대효과:

U-boot의 커맨드 모드로 진입한 다음 아래의 명령을 수행한다.

```
SMDK6400 # setenv bootdelay 0
SMDK6400 # saveenv
SMDK6400 # reset
```

U-boot 작동시 사용자 입력대기를 더 이상 기다리지 않는다. 따라서 빠르게 부팅을 진행할 수 있다. 그러나 아예 커맨드 모드로 들어갈 수 없는 것은 아니다. 디버그 포트가 RS-232C 기반의 시리얼 포트이므로 U-boot 타이틀 메시지 출력시 키를 미리 눌러두면 커맨드 모드로 진입이 가능하다.

2.2 불필요한 초기화 루틴제거

U-boot 수행의 가장 중요한 임무는 커널을 부팅시키는데 필요한 하드웨어 초기화와 커널을 플래쉬에서 일반 메모리로 복사하는 일이다. 따라서 이와 관련한 작업 외의 다른 모든 작업들은 어차피 커널에서 수행하게 되므로 모두 제거한다.

적용 및 기대 효과:

직접 U-boot 소스코드를 수정한다. CPU, DRAM 초기화 루틴과 커널 복사 루틴을 제외한 나머지를 부팅시 건너 뛰게 만든다. 단, 디버그 포트의 초기화는 수행한다.

비디오나 시리얼포트 등을 초기화 하지 않으므로 약간의 부팅 시간 단축효과가 예상된다.

2.3 커널 이미지 비압축로드

커널 이미지는 최초 커널 컴파일하여 빌드 시 그림1의 왼쪽과 같이 압축형태의 커널 이미지와 그것의 압축을 해제할 수 있는 루틴이 결합되어 있다. 이것을 그림1의 오른쪽 그림과 같이 압축을 하지 않으면 커널의 이미지 크기가 커져 로드 시간이 길어지지만 gz 형태의 압축을 해제하는 시간을 건너 뛸 수 있어 결과적으로 부팅시간을 단축하게 된다. 다만, 플래쉬의 용량이 허용되지 않는 경우는 사용할 수 없는 기술이다.

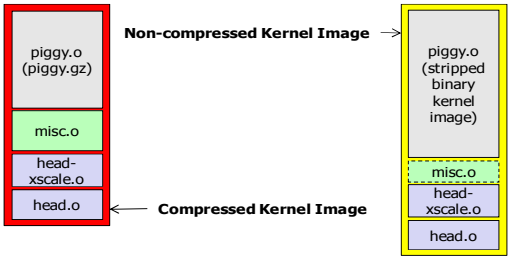

그림 1 리눅스 커널 이미지의 형태

적용 및 기대 효과:

커널 이미지 빌드 시 아래와 같이 컴파일 한다. 그리고 복사한다.

```
fb_host$ make ARCH=arm CROSS_COMPILE=arm-linux- Image
fb_host$ cp arch/arm/boot/Image /tftpboot/Image
```

커널의 부팅 이전의 시간은 커널을 메모리로 복사하는 시간과 그것의 압축을 푸는 시간의 합이다. 결국 커널 이미지 자체는 더 커지지만 압축을 푸는 시간을 없앰으로써 부팅 시간을 줄일 수 있다.

2.4 최적화된 복사 루틴 사용

U-boot는 최초 플래쉬에 저장되어 있다가 자체 복사 루틴에 의해서 일반 메모리로 옮겨진다. 그 후 적절한 초기화를 거친 다음 최종 명령으로 플래쉬 내의 커널 이미지를 메모리로 복사한다. 하드웨어 의존도를 낮추기 위해 복사의 측면에서 호환성이나 유연성 때문에 U-boot는 특정 하드웨어나 보드에 관계없이 약간의 하드웨어 지원만으로 동작해야 한다. 그러기 위해서 각 하드웨어에서 가장 속도가 잘 나오는 특정 방법은 쓰지 못하고 일반적인 C언어 상의 포인터를 통한 복사를 수행한다. 문제는 이러한 방법이 각 메모리에 대한 최적화된 방법이 아니라는 것이다.

적용 및 기대 효과:

S3C6400이 탑재된 SMDK6400 보드의 경우는 CPU 코어가 Idle 상태라는 가정하에 DMA 채널을 이용하여 복사하면 더 빠른 속도로 복사를 할 수 있게 된다. 단 DMA 채널을 사용하려면 인터럽트 루틴이 활성화되고 일반 메모리에 존재하는 스택을 사용할 수 밖에 없으므로 U-boot 자체의 복사에는 활용되기 어렵고, U-boot에서 커널 이미지를 복사하는 부분에 사용이 가능하다. 일반 복사 루틴에서 DMA 처리 루틴으로 바뀌는 경우 메모리 복사 속도를 4배 정도 높일 수 있는 것으로 예상된다. 따라서 2초 정도 걸리는 커널 복사

루틴 점유 시간을 0.5초 이내로 단축시킬 수 있을 것으로 전망된다.

Ⅲ. 부팅 개선방법 II: 초기화 스크립트 단계

일단 커널 부팅 과정에서 초기화 과정이 완료되면 루트 파일시스템을 마운트하고 특정 프로세스인 init을 실행한다. init은 모든 프로세스들의 부모이면서 동시에 fork & exec 기법이 아닌 커널에서 직접 실행되는 유일한 프로세스이다. POSIX 표준에 의하면 init 프로세스는 모든 프로세스들의 부모이므로 소멸시 다른 모든 프로세스들 보다 나중에 소멸될 의무가 있다. 결국 init 프로세스의 소멸은 시스템 재시작이나 전원끄기가 된다. init은 이후 구현에 따라 달라지지만 리눅스에서 사용되는 init의 경우에는 자신의 초기화 과정을 수행하고 나면 "/etc/inittab"파일을 읽어서 이후 행동들을 준비한다. 원칙적으로 inittab에는 초기화 수행시 실행할 파일의 이름과 각 런레벨시에 어떤 대응을 할지가 기술되어 있다.

3.1 이진화 기반 스크립트 대체 기술 사용

전술한 바와 같이 init은 시스템의 초기화를 /etc/inittab을 통해 쉘스크립트로 초기화수행을 한다. 쉘스크립트는 사람이 직접 읽을 수 있고 즉시 수정이 가능하므로 지금까지의 여러 리눅스에서 환영을 받아 왔다. 그러나 쉘스크립트는 매 라인을 직접 읽어 해석을 해야 하는 인터프리터 방식이다. 따라서 쉘스크립트 안에서는 의미없이 주석문을 이용한 것도 문제가 될 수 있다. 물론 그런 부분들을 제거해도 되지만 그렇게 되면 쉘스크립트가 매우 난해해 진다.

또한가지 더 큰 문제는 쉘스크립트가 쉘기반이다 보니 쉘에서 어떤 명령어를 수행하기 위해서는 반드시 fork & exec를 사용해야 한다는 것이다. 요즘은 기법이 발달하여 프로세스를 fork 한다 해도 직접적으로 메모리 복사가 되는 것은 아니다. 그러나 커널 내부에서 프로세스를 유지관리하기 위해 사용되는 task_struct로 표현되는 일련의 구조체들은 복사 및 유지되어야 한다. 성능이 좋은 데스크탑형이나 서버형의 PC에서는 이 부분은 크게 문제되지 않지만 임베디드 시스템에서는 문제가 된다.

이런 단점들에도 불구하고 지금까지 쉘스크립트를 고수한 까닭은 문제가 생겼을 경우 수정이 쉽고 만들기가 간편하기 때문에 개발자들이 선호하기 때문이다. 그러나 빠른 부팅은 이제 고객의 요구사항이며 고객이 직접 쉘스크립트를 수정하는 것이 아니기 때문에 가장 빠른 형태인 바이너리 형태로 바뀌어야 한다.

적용 및 기대 효과:

초기화 과정을 이진화 하려면 먼저 초기화 과정의 전체 흐름을 파악하고 각 단계의 작동을 최대한 단순화하는 것이 중요하다. 각각의 단계에 대한 흐름은 보드마다 다를 수 있다.

그림2는 SMDK6400단말의 초기화과정을 이진화할 수 있도록 단순화한 것이다. 이 과정에서 init은 더 이상 /etc/rc.d/rc.sysinit이나 /etc/rc.d/rc 파일에 의존하지 않는다.

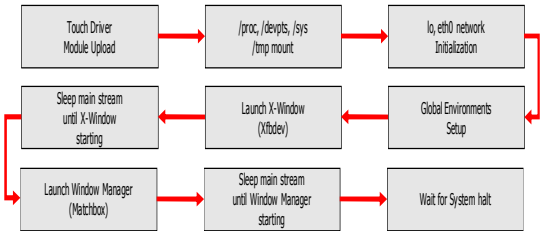

그림 2 SMDK6400의 단순화된 초기화 과정

이들 각각의 과정을 이진화하기 위해서는 각 모듈 별로 이진화가 되어 있어야 한다. 본 연구에서는 이진화 스크립트에 필요한 각 모듈들의 기능을 위해 7개의 함수를 만들어 이진화하였다.

각 모듈별 이진화 작업이 완료 되면 처리 순서에 따라 이것을 처리하는 기술이 필요하다. 일단 각 부분들에 대한 모듈화가 이루어지면 이것을 구조체의 집합으로 묶어서 이진화 스크립트 실행시 각각의 모듈들을 차례대로 실행시킨다. 이렇게 구성된 구조체 집합은 메인 루틴에 의해 실행될 때 비로소 이진화 스크립트로 동작한다.

3.2 init 프로세스 경량화

임베디드 리눅스의 부팅 과정에서 커널 부팅이 완료되면 1번 프로세스인 init 프로세스를 부르게 된다. 리눅스 커널은 1번 프로세스가 정상적이든 비정상적이든 종료될 때까지 커널의 메인 루틴은 대기상태에 있게 된다. 따라서 실제 시스템 운용중 커널과 관련하여 일어나는 모든 일들은 서비스 루틴들이다.

응용 프로그램 측면에서 init 프로세스의 중요성은 실제 사용자에게 서비스를 제공해주는 것은 아니지만 모든 프로세스의 부모라는 측면에서 매우 높다. 이러한 측면에 볼 때 init가 가져야 하는 특징들을 정리하면 다음과 같다.

가. init 최초 구동 시 빠른 부팅을 지원하기 위해서는 가급적 작은 규모의 코드이어야 한다. 리눅스 커널 부팅 완료 후에 커널은 /sbin/init을 커널 API인 execve를 통하여 메모리로 읽어들이게 된다. 이때 init의 크기가 크면 로드하는데 시간이 많이 걸릴 것이다.

나. init은 특성 상 fork를 매우 많이 하는 프로세스이다. fork 자체가 프로세스가 차지하는 메모리를 복사하는 것은 아니나 사용 메모리가 많을수록 커널 내부의 제어 블록들은 그 크기가 커지게 되고 이들은 프로세스 메모리와는 달리 복사된다. 또한 exec를 수행할 경우 전체의 메모리를 flush하고 새로운 내용으로 덮어쓰게 된다. 이때도 init의 메모리 차지 비중이 커지면 flush하는데도 시간이 많이 소요된다.

다. init 프로세스는 모든 프로세스의 부모이므로 다른 프로세스가 종료될 때 시그널 및 기타 처리를 수행한다. 임베디드 시스템에서 프로세스들의 수를 늘리는 것은 좋은 방법이 아니지만 다수의 프로세스들이 존재할 수 있으므로 이들을 처리하는데 많은 자원이 소모되어서는 곤란하다.

적용 및 기대 효과:

/sbin/init을 교체한다. 본 논문에서는 /sbin/init과 simplebinrc.sysinit을 결합한다. 이진화 스크립트와 /sbin/init을 결합하는 것은 init이 스크립트를 사용하여 시스템을 초기화하므로 연관성이 높다. 따라서 init을 경량화 시 처음부터 이진화 스크립트와 결합하여야 한다. 이진화 스크립트를 init과 결합을 하면 첫째, 부팅시 busybox를 로드하지 않아도 된다. busybox는 쉘 기능에 있어서 매우 편리한 도구이지만 모든 기능을 다 구현하다 보니 크기가 매우 크다. 일례로 SMDK6400 보드에 구현된 busybox는 1.7Mbyte 정도가 된다. 이러한 크기를 메모리로 로드하는 데만도 초당 2MB/sec 정도의 플래쉬 메모리에서는 1초에 가까운 시간이 허비된다. 따라서 init을 경량화 하는 문제는 필연적이다.

Ⅳ. 실험결과

본 논문에서는 임베디드 시스템 개발보드로 Naitec PC100 보드를 이용하여 부트로더 단계와 초기화 스크립트 단계에 대해 각각 적용하여 결과를 측정하였다. 실험에 사용한 개발보드의 하드웨어 및 소프트웨어 스팩은 Naitec PC100 board: S5PC100@667MHZ, 256Mbytes DDR2 SDRAM, 256Mbytes NAND Flash, 부트로더 uboot 1.1.6, 커널 버전 Linux 2.6.X 이며, 그 결과는 표1과 같다.

표 1 빠른 부팅 실험결과

	적용 전	적용 후
부트로더 단계	3.42초	3.37초
초기화 스크립트 단계	9.36초	1.17초
종합	12.78초	4.54초

또한 상용제품으로 제공되는 GPH 게임기 CAANOO, 조선단말 CX200 제품에 대하여 부트로더 단계 개선방법과 초기화 스크립트 단계 개선방법을 적용한 실험하였다. 그 결과는 다음과 같다.

- GPH 게임기CAANOO:

하드웨어 및 소프트웨어 스팩: Pollux ARM@533MHZ, 128Mbytes DDR SDRAM, 128Mbytes NAND Flash, 부트로더 uboot 1.1.6, 커널 버전 Linux 2.6.X
실험결과: 개선방법 적용 전 20초, 개선방법 적용 후 11초

- 조선단말 CX200:

하드웨어 및 소프트웨어 스팩: S3C6410@533MHZ, 256Mbytes DDR SDRAM, 256Mbytes NAND Flash, 부트로더 uboot 1.1.6, 커널 버전 Linux 2.6.X
실험결과: 개선방법 적용 전 40초, 개선방법 적용 후 14초・

Ⅴ. 결 론

본 논문에서는 빠른 부팅 개선을 위해 전원인가 시 최초로 실행되는 부트로더 단계와 초기화 스크립트 단계에 적용했다. 부트로더 단계에서 입력 대기 시간 제거, 불필요 초기화 루틴제거, 커널 이미지 비압축 로드, 최적화된 복사 루틴 사용 등을 적용하여 부팅 개선을 했다. 또한 초기화 스크립트 단계에서 이진화 기반 스크립트 대체 기술 사용, init 프로세스 경량화 등을 적용하여 부팅 개선을 했다.

본 연구의 결과는 임베디드 리눅스 기반 시스템 분야에서 다양한 제품들의 부팅 시간 단축을 위해 사용될 수 있다. 특히, 이진화 스크립트를 사용하여 부팅 시간을 줄이는 방법은 임베디드 시스템 같은 성능 자원이 한정적인 곳에서는 그 효과가 극대화될 수 있다. 그러나, 한번 이진화 스크립트가 작성되면 그것을 바꾸기 쉽지 않고 이진화 스크립트 자체가 C언어로 작성되었으므로 누군가가 그것을 수정하거나 추가하려면 해당 언어에 대한 지식이 있어야만 한다. 이런 문제를 해결하기 위해서는 시스템 초기화 스크립트를 이용하여 이진화 스크립트를 제작할 수 있는 도구가 필요하게 된다. 따라서 앞으로 임베디드 리눅스에 최적화된 이진화 스크립트 제작도구 연구와 빠른 부팅 개선을 위해 부팅과정에 있는 커널 단계, 파일 시스템 단계, 공유라이브러리 단계, 응용프로그램 단계에 대한 성능개선이 요구된다.

참고문헌

[1] 도인환, “임베디드 리눅스 시스템에서 하이버네이션 기반 부팅 방식 구현”, 한국컴퓨터정보학회지 제16권 제5호 통권 제86호, p23-p31, 2011

[2] 박세진, 송세환, 박찬익, 송재환, “개선된 스냅샷 부트를 이용한 임베디드 리눅스의 빠른 부팅 기법”, 정보과학회논문지 제14권 제6호 (2008년 8월) p594-p598, 2008

[3] 남상엽, 강민구, 정태경, 구제길, 이강현, “IPTV 구성 및 응용”, 상학당, p297-p393, 2009

[4] 신광무, 박성호, 정기동, “임베디드 시스템에서 리눅스의 빠른 부팅”, 제32회 한국정보과학회 추계학술발표회 논문집 vol.32, No.2(I), p853-p855, 2005