플래쉬 메모리 내에 상주 가능한 경량 리눅스 운영체제 구현

장 승 주*

Implementation of Light Weight Linux O.S on the Flash Memory

Seung-Ju Jang*

본 논문은 2007년도 동일문화장학재단 연구비 지원에 의하여 이루어졌음.

요 약

최근 임베디드 시스템에 대한 많은 연구들이 진행 중이다. 임베디드 시스템은 점점 소형화 추세로 가고 있다. DOM(Disk On Module)저장장치는 공간에 제한이 있는 응용프로그램이나 모바일 등의 기기에 사용할 수 있다. 본 논문에서는 DOM(Disk On Module) 메모리를 사용하여, 리눅스 기반의 커널을 탑재하고, DOM 메모리 만으로 시스템이 구동될 수 있도록 한다. DOM(Disk On Module) 메모리의 용량 제한으로 인하여 소형 운영체제가 필수적이다. 이를 위해 본 논문은 기존의 리눅스 운영체제를 DOM 환경에 적합하도록 경량화시켜서 설계하였다. 리눅스 운영체제를 경량화한 후, DOM(Disk On module)에 부트 로더의 한 종류인 LILO를 설치하여 DOM(Disk On module)메모리 상에서 새롭게 설계된 경량 리눅스 운영체제가 일반 리눅스 운영체제처럼 부팅될 수 있게 만들어 준다. 본 논문은 일반 리눅스 PC와 성능을 비교하는 실험을 수행하였다.

ABSTRACT

Many people is studying the embedded system. The embedded system becomes a small size device. The DOM memory is using in the mobile device and small size devices. This paper proposes light-weighted Linux O.S that is running onto the DOM memory. The embedded system with the DOM must have a light-weigthed O.S due to the memory space restriction. This paper designs light-weigthed Linux O.S for the DOM memory. The new designed LILO boot loader boots the new designed light-weigthed Linux O.S as a normal Linux O.S. This paper experiments comparing the designed new light-weigthed Linux O.S with a Linux PC.

키워드

DOM, 경량 운영체제, 리눅스 O.S

Ⅰ. 서 론

DOM(Disk On Module)으로 알려진 IDE 타입의 FDM(Flash Disk Module)은 반도체를 이용한 IDE 타입의 저장 매체로서 E-IDE 하드디스크 드라이브를 완벽하게 대체할 수 있는 차세대 저장 매체이다. DOM(Disk On Module)을 사용하는 이유는 다양하지만 가장 큰 이유로는 하드디스크에서 흔히 발생하는 Seek Error가 발생하지 않는다는 점에서 사용한다. 충격이 가해지고 흔들리는 환경에서도 DOM(Disk On Module)은 전혀 Seek 에러

* 동의대학교 컴퓨터공학과 교수

접수일자 : 2007. 6. 28

가 발생치 않는다. 또한 DOM(Disk On Module)은 크기가 HDD에 비해 작으며 예를 들어 영하의 날씨와 같은 가혹한 환경에서도 문제없이 동작하기 때문에 DOM(Disk On Module)은 공간에 제한이 있는 응용프로그램이나 모바일 응용프로그램에 최적의 해법이다. 여기서 말하는 DOM(Disk On Module)은 플래시메모리의 일종이다[1,12,13,14].

플래시메모리는 기존의 기억장치인 하드디스크를 대체할 것으로 주목받고 있는 차세대 기억장치로서 집적 회로로 구성된 비휘발성 메모리이다. 최근 플래시메모리의 집적도와 입출력 성능이 계속 개선되고 있어 이동 컴퓨터의 보조기억 장치로 뿐만 아니라 일반 컴퓨팅 시스템의 저장 매체로까지 사용할 수 있게 되었다. 플래시 메모리는 전기적으로 입출력을 수행하기 때문에 읽기 성능은 D램과 거의 같은 수준이며 쓰기 속도도 하드디스크에 비해 수백 배 이상 빠르다. 그리고 이 저장 매체는 같은 크기의 D램보다 그 크기가 약 30% 작으면서도 내충격성과 고장률 측면에서 훨씬 우수하기 때문에 대용량화가 가능하다. 또한 플래시메모리에 존재하는 데이터는 주기억장치로 읽어 들여 질 필요 없이 직접 플래시메모리 공간상에서 입출력이 가능하다. 이러한 특성 때문에 플래시메모리가 기존의 디스크 기반 저장시스템의 저장 매체를 대체하는 경우에 시스템의 입출력 성능을 획기적으로 향상시킬 수 있다[1,2,9,10,11].

본 논문에서는 리눅스 운영체제를 일반 E-IDE대신, DOM(Disk On Module)을 사용할 수 있도록 한다. 본 논문에서 제안하는 DOM 메모리 내의 경량 리눅스 운영체제 구현 연구 결과는 소형 전자 기기에 유용하게 활용 가능하다. 본 논문에서 구현한 시스템 환경은 인텔 펜티엄 III 850MHz CPU, 256Mbyte RAM 이고, 리눅스 운영체제는 RedHat Linux 9.0 - Kernel Version 2.4.28 이다. 본 논문에서는 IDE방식의 HDD(Hard Disk)에 기본 옵션의 리눅스 운영체제를 설치한다. DOM(Disk On Module) Memory를 사용하여, 리눅스 기반의 커널을 탑재하고, 그 DOM(Disk On Module) Memory만으로 시스템이 구동될 수 있도록 환경을 구축 한다. 환경을 구축하는데 있어 DOM(Disk On Module) Memory의 용량의 제한으로 인하여 부득이 하게 리눅스 커널을 최대한 간소화 하는 작업이 필요하다. 리눅스 커널을 최대한 간소화 하기위해서는 먼저 리눅스 디렉터리 구조를 분석하고, 그 디렉터리들이 어떤 역할을 하는지 분석한다.

디렉터리의 구조를 분석하였으면, 꼭 필요하지 않은 디렉터리들은 삽입하지 않고, 필요한 디렉터리를 정리한 후 DOM(Disk On Module)에 추가, 설정을 한다. 이러한 작업은 DOM(Disk On Module)을 사용하기 위해서 리눅스 커널의 최적화 작업이라고 할 수 있다. 커널의 최적화 작업을 마무리하고, 커널을 DOM(Disk On Module)에 복사한다. DOM(Disk On module)에 부트로더의 한 종류인 LILO를 설치하여 DOM(Disk On module)이 일반 리눅스 PC처럼 부팅을 할 수 있게 만들어 준다. 마지막으로 DOM(Disk On Module)에 설치되어 있는 리눅스 운영체제의 구동이 잘 되는지 실험 및 성능 측정으로 본 논문을 마친다.

논문의 구성은 다음과 같다. 2장에서는 관련연구를 언급한다. 3장은 경량 리눅스 운영체제의 설계 및 개발 내용을 언급한다. 4장에서는 실험 내용에 대해서 언급한다. 5장에서 본 논문에 대한 결론을 내린다.

II. 관련연구

본 논문과 관련된 관련 연구에 대해서 살펴본다. 본 논문에서 제안하는 방식은 DOM 메모리 내에서 부팅 가능한 임베디드 운영체제이다. 임베디드 운영체제는 다양한 영역에서 적용되고 있다. 임베디드 운영체제들의 공통적인 특징은 이들이 실시간 운영체제 시스템이라는 것이다. 많은 임베디드 시스템들은 경량화 와 실시간성에 무게를 두고 개발되어왔다. 실시간 요구사항은 하드 리얼타임과 소프트리얼타임으로 분류할 수 있지만 정해진 시간 내에 태스크를 실행하기 위하여 선점형 프로세스 스케줄링 방식을 채용하고 비동기 시그널 처리를 지원한다. 또한 이들 운영체제들은 개발자의 편의를 위해 임베디드 운영체제들의 또 다른 특징은 경량화이다[1,2,3,4,5,15,16,17].

경량화 역시 임베디드 시스템이 갖는 제약적 하드웨어 환경 요소로 인한 특징이다. 일부 임베디드 시스템들은 저전력 소비를 위한 알고리즘을 내재하고 있다. 또한 최소한의 연산을 수행하기 위한 내부구조를 지원한다. 기존에 연구된 임베디드 운영체제는 다음과 같다[6,7,8].

표 1. 임베디드 시스템용 운영체제
Table 1. Embedded System Operating System

제품명	개발사
VxWorks, pSoS	Wind River
VRTX	Ready Systems
MicroC/OS-II	Micrium INC.
RTLinux	FSM Labs
Nucleus	Accelerated Technology
Windows CE.NET, XP Embedded, Mobile 2003	Microsoft

그러나, 기존의 임베디드 운영체제도 PC와 마찬가지로 보조기억장치내에 프로그램 형태로 존재하면서 주기억장치 내에서 동작하는 구조를 갖고 있다. 그러나, 앞으로 사용될 많은 임베디드 시스템은 DOM을 가진 구조가 될 것이다. 따라서 기존의 임베디드 운영체제 방식으로는 한계가 있다. 따라서, 본 논문에서는 이러한 방식을 극복한 경량 리눅스 운영체제를 제안한다.

Ⅲ. 설계 및 개발 내용

본 절에서는 본 논문에서 제안하는 경량 리눅스 운영체제의 설계 내용에 대해서 언급한다. 다음 그림 1.은 본 논문에서 설계하고자 하는 경량 리눅스 운영체제를 위한 시스템 구조를 나타낸다.

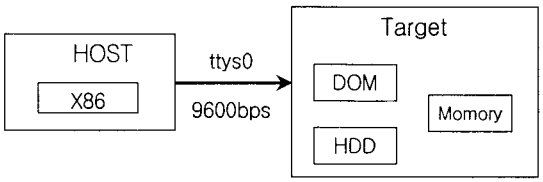

그림 1. 경량 리눅스 운영체제를 위한 컴퓨터 시스템 구조
Fig. 1. System Architecture for Lightweight O.S

그림 1.은 DOM 메모리를 이용한 Redhat 9.0 은영체제 구축 시스템 구조로써 HOST PC와 Target으로 나누어지며, DOM은 Target 시스템에 설치되며, HOST와 Target연결은 시리얼 포트를 사용 minicom 에뮬레이터를 사용한다. 본 논문에서 제안하는 경량 리눅스 운영체제 설계과정은 다음과 같다.

3.1 DOM에 리눅스 커널 설계를 위한 주요 디렉터리 구조

기존의 리눅스 커널 디렉토리 구조에서 본 논문에서 설계하고자 하는 경량 리눅스 운영체제를 위한 디렉토리 구조를 설계한다. 아래 그림 2.는 기존의 리눅스 운영체제 디렉토리 구조를 이용하여 DOM 메모리 내에서 필요한 최소한의 디렉토리 구조를 보여준다.

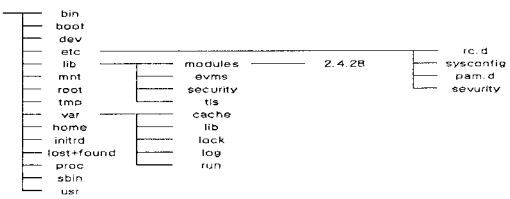

그림 2. 경량 리눅스 운영체제 설계를 위한 디렉토리 구조
Fig. 2. Design Directory Structure for Lightweight O.S

그림 2.는 리눅스 운영체제가 구동될 때 최소한의 필요한 디렉터리 구조를 보여준다. 위와 같은 구성으로 제안된 시스템의 DOM 메모리 내에 Redhat Linux 9.0 커널을 이용하여 경량 리눅스 시스템 환경을 구축하였다. 각 디렉토리의 기능은 다음과 같다.

/bin 폴더 - 시스템 운영을 위해 필요한 명령어들이 저장되어 있는 폴더이다. 일반적으로 부팅에 필요한 명령어와 시스템의 계정 사용자들이 사용할 수 있는 일반적인 명령어들이 저장되어 있다.

/boot 폴더 - Linux boot에 필요한 booting 지원 파일들이 위치하는 디렉토리

/dev 폴더 -시스템의 각종 디바이스들에 접근하기 위한 디바이스 드라이버들이 저장된 디렉토리

/etc 폴더 - 시스템에 관한 각종 환경 설정 파일과 디렉토리를 가짐

/lib 폴더 - 프로그램들의 각종 라이브러리 파일들이 위치하는 디렉토리

/root 폴더 - 시스템 user의 홈 디렉토리이다.

/proc 폴더 - 시스템의 각종 프로세서, 프로그램 정보 그리고 하드웨어적인 정보를 저장

/sbin 폴더 - Linux kernel에 필요한 module 설정 파일이 위치하는 디렉토리

/usr 폴더 - Linux에서 사용하는 모든 어플리케이션 및 시스템 파일이 위치하는 디렉토리

3.2 경량 리눅스 커널 설계

본 논문에서 제안한 경량 리눅스 운영체제 커널을 제작하기 위하여 위 그림 2.에서 제안하는 디렉토리 구조의 기본으로 커널 이미지 제작을 한다. 리눅스 운영체제의 "cp -a" 명령어를 사용하여 리눅스 운영체제 관련 파일 복사 시에 파일 관련 정보 권한 변경이 없도록 하여야 한다.

경량 리눅스 운영체제 커널 이미지 구성을 위해서 각각의 소스 코드에서 생성된 각 object 파일별로 필요한 라이브러리 확인할 때 ldd 명령을 사용한다. 이 명령어의 사용으로 각 object에서 필요한 라이브러리를 점검한 후 실제 필요한 라이브러리 파일을 복사한다. 복사된 라이브러리는 각 object가 수행하는데 필요하다.

경량 리눅스 운영체제 커널 이미지를 생성하기 위하여 가상 파일 시스템을 제작한다. 가상 파일 시스템을 제작함으로써 "chroot(1M)" 리눅스 명령어를 사용하여 만들어진 가상 파일 시스템을 하나의 새로운 / 디렉토리로 하여 복사한 명령어나 라이브러리가 올바르게 동작이 되는지 확인하는 작업을 쉽게하도록 해준다.

DOM 메모리 내에서 동작 가능한 경량 리눅스 운영체제 설계를 위한 과정은 다음과 같다.

O 가상 파일 시스템 제작
O 필요한 파일 및 library 복사
O 필요한 소스 파일을 컴파일 하여 삽입
O 이미지 만들기

3.2.1 가상 파일 시스템 구축

가상 파일 시스템을 제작함으로써 chroot를 사용하여 만들어진 가상 파일 시스템을 하나의 새로운 / 로 하여 복사한 명령어나 라이브러리가 올바르게 동작이 되는지 확인하는 작업을 편하게 해준다.

(1) chroot 명령어가 실행되기 위해서 수반되어야 할 작업

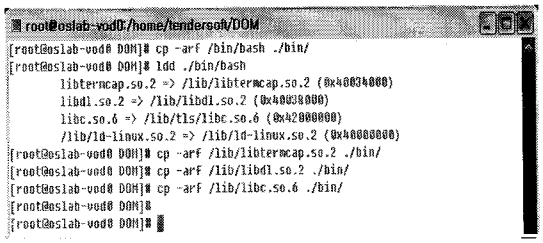

그림 3. bash object 파일을 복사하는데 이행되는 과정

Fig. 3. Copy Procedure of bash Object Files

그림 3.은 bash object 파일을 복사하는 과정을 나타내는 과정으로써 먼저 object파일을 복사한 다음 그에 필요한 라이브러리를 ldd 명령을 사용하여 확인한 다음 라이브러리를 복사를 한다. 여기서 주의해야 할 점은 라이브러리가 심볼릭 링크가 되어 있다면 그 심볼릭 링크까지 복사를 해주어야 한다.

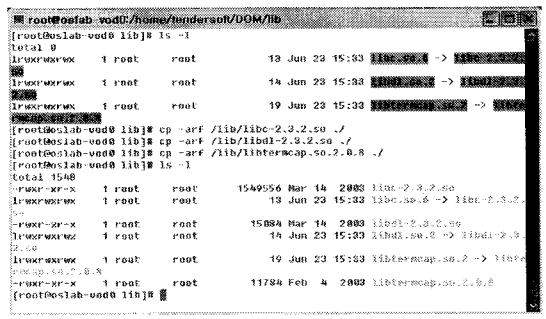

그림 4. 라이브러리의 심볼릭 링크를 복사하는 과정

Fig. 4. Copy Procedure of Library Files

그림 4.와 같이 bash와 관련된 라이브러리 복사가 끝났다면 복사를 한 최상위 root 폴더에서 chroot ./를 사용하여 bash가 실행 되게 만들면서 각각의 오브젝트들을 복사하여 올바르게 동작을 하는지 확인을 하면 된다.

(2) 필요한 파일 복사 및 라이브러리 복사

필요한 파일 및 라이브러리를 복사한다. 복사할 때 주의 사항은 다음과 같다. cp -a 옵션을 사용하여 복사시 파일관련 정보 권한 변경이 없도록 하여야 한다. 각 object 파일별로 필요한 라이브러리 확인할 때 ldd 명령을 사용한다. /bin 디렉토리에서 복사되어야할 파일은 다음과 같다.

awk, basename, bash, cat, chgrp, chmod, chown, cp, cut, date, dd, df, dmesg, dnsdomainname, doexec, domainname, echo, egrep, env, false, fgrep, gawk, grep, hostname, igawk, ipcalc, link, ln, login, ls, mkdir, mknod, mount, mv, netstat, nice, nisdomainname, pgawk, pwd, rm, rmdir, sed, sh, sleep, sort, stty, su, sync, touch, true, umount, uname, unlink, usleep, vi, ypdomainname

그림 5. /bin 디렉토리에 복사되는 파일들
Fig. 5. Copy Procedure of /bin directory

다음 그림 5.는 /bin 디렉토리 내에 파일들을 복사하는 과정을 보여준다.

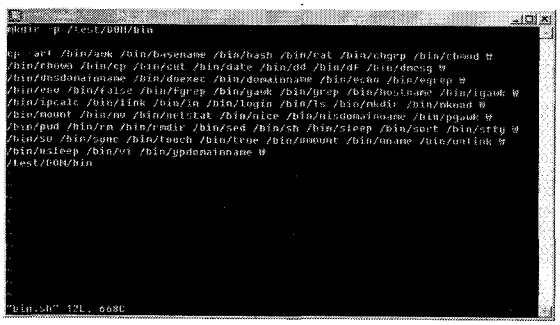

그림 6. /bin 디렉토리 내에 파일들을 복사하는 과정
Fig. 6. Copy Procedure of files inside /bin directory

다음 그림 7.은 /boot 디렉토리 내에 복사하게 될 파일들을 보여준다.

grub*, System.map-2.6.17-1.2157_FC5,
initrd-2.6.17-1.2157_FC5.img,
config-2.6.17-1.2157_FC5,
vmlinuz-2.6.17-1.2157_FC5

그림 7. /boot 디렉토리 내에 복사할 파일들
Fig. 7. Copy Files of inside /boot directory

다음 그림 8.은 /bin 디렉토리 내에 파일들을 복사하는 과정을 보여준다.

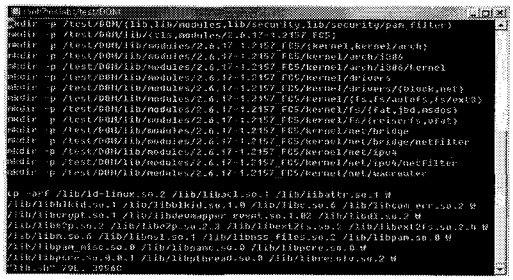

그림 8. /boot 디렉토리 내에 파일들을 복사하는 과정
Fig. 8. Copy Procedure of files inside /boot directory

다음 그림 9.는 /lib 디렉토리 내에 복사하게 될 파일들을 보여준다.

modules*, security*, tls*, ld-linux.so.2, libacl.so.1, libattr.so.1, libblkid.so.1, libblkid.so.1.0, libc.so.6, libcom_err.so.2, libcrypt.so.1, libdevmapper-event.so.1.02, libdl.so.2, libe2p.so.2, libe2p.so.2.3, libext2fs.so.2, libext2fs.so.2.4, libm.so.6, libnsl.so.1, libnss_files.so.2, libpam.so.0, libpam_misc.so.0, libpamc.so.0, libpcre.so.0, libpcre.so.0.0.1, libpthread.so.0, libresolv.so.2

그림 9. /lib 디렉토리 내에 복사할 파일들
Fig. 9. Copy Files of inside /lib directory

/lib 디렉토리에 필요한 파일들을 복사하는 과정은 앞에서 설명한 다른 디렉토리에서와 같다.

agetty, arp, arping, badblocks, blockdev, consoletype, debugfs, depmod, dumpe2fs, e2fsck, e2image, e2label, ether-wake, findfs, fsck, fsck.ext2, fsck.ext3, getkey, grub, grub-install, grub-md5-crypt, grub-terminfo, grubby, halt, hwclock, ifconfig, ifdown, ifup, init, initlog, insmod, insmod.static, ip, ipmaddr, iptunnel, killall5, klogd, kudzu, ldconfig, lsmod, mii-tool, mingetty, mke2fs, mkfs.ext2, mkfs.ext3, modinfo, modprobe, nameif, netreport, pam_console_apply, pam_tally, pam_timestamp_check, pidof, plipconfig, poweroff, ppp-watch, reboot, resize2fs, rmmod, route, runlevel, service, setsysfont, shutdown, slattach, sulogin, sysctl, syslogd, telinit, tune2fs, unix_chkpwd

그림 10. /sbin 디렉토리 내에 복사할 파일들
Fig. 10. Copy Files of inside /sbin directory

위와 같은 과정에 의하여 필요한 리눅스 운영체제 동작 환경이 구축이 된 것이다. 이러한 기본적인 동작 환경이 설계되고 난후에 실제 커널의 이미지 파일을 생성해야 한다. 커널 이미지 파일 생성 과정은 다음과 같다.

3.2.2 64Mbyte 리눅스 운영체제 커널 이미지 만들기

경량 리눅스 운영체제 커널 이미지를 생성하기 위하여 아래와 같이 리눅스의 "dd" 명령어를 사용한다. 아래는 "dd" 명령어의 사용법이다.

```
dd if=/dev/zero of=szips.img bs=1k count=[용량]
```

위 "dd" 명령어에서 64Mbyte 경량 리눅스 이미지를 만들기 위해서는 bs의 1k와 count 값인 [용량]을 곱한 값이 되도록 하면 된다. 이 기능을 이용하여 최소한의 기능을 가진 경량 리눅스 운영체제 커널 이미지를 만들 수 있다.

3.2.3 리눅스 커널 이미지에 파일 시스템 복사

리눅스의 "mkfs.ext3 -N 23000" 명령어를 사용하여 리눅스 이미지 내에 리눅스 ext3 파일 시스템을 만들어 준다. 이 파일 시스템의 생성으로 설계된 리눅스 운영체제가 동작 중일 때 정상적으로 파일을 사용가능하도록 한다. 여기서 -N 23000은 파일 시스템 내에 inode의 갯수를 지정하는 기능이다. 이렇게 생성된 ext3 파일 시스템에서 전체 파일을 "cp -a" 명령어를 사용하여 만들어 놓은 경량 리눅스 이미지로 복사하면 최종 리눅스 커널 이미지가 완성된다. 이렇게 생성된 ext3 파일 시스템은 임의의 폴더에 마운트 할 수 있는데, 마운트 하는 방법으로는 "mount [이미지명] [마운트지점] -o loop" 명령어를 사용할 수 있다. 이런 과정을 통해서 최종 경량 리눅스 운영체제 커널 이미지 생성을 마무리할 수 있다. 생성된 경량 리눅스 운영체제 커널 이미지를 실제 DOM 메모리 내에 동작이 가능하다.

Ⅳ. 실험

구현된 경량 리눅스 운영체제 환경을 실험하기 위하여 minicom을 이용한다.

4.1 minicom 설정

타켓시스템의 com단자와 Host 시스템의 com1 단자에 연결하여 시리얼 통신을 위한 환경을 구축한다. minicom 명령의 -s 옵션을 사용하여 minicom을 설정한다. 여기서 시리얼 통신과 관련된 설정을 한다. 세 번째 환경인 Serial port setup을 선택하여 전송속도 등을 설정한다. 설정이 정상적으로 완료되면 터미널 화면에 타켓 시스템의 초기화 화면 그림 11.과 같이 뜨면 minicom 설정이 올바르게 된 것이다.

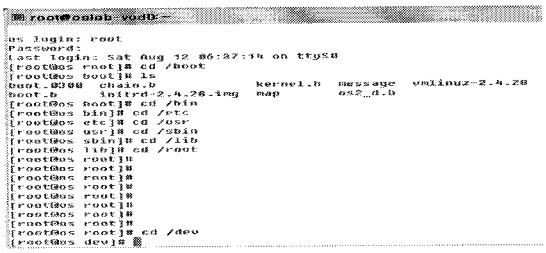

그림 11. minicom으로 타켓 시스템에 접속한 화면
Fig. 11 Connecting Screen to the Taget System using minicom

4.2 메모리 성능 측정 프로그램 작성 및 테스트

본 논문에서 제안하는 경량 리눅스 운영체제에 대한 실험을 위하여 메모리 성능 측정을 위한 프로그램을 작성 및 테스트하였다. 또한 성능 벤치마크 프로그램을 이용하여 경량 리눅스 운영체제를 실험하였다. 메모리 성능 측정 프로그램은 다음 그림 12.와 같다.

```
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <stdlib.h>
#define BUFSIZE 2048
#define MAX 999999
int main() {
  clock_t start;
  int i, j;
  float diff;
  char testTmp[BUFSIZE], buf[BUFSIZE];
  start = clock();
  for(i=1; i <= MAX; i++) {
    for(j=0; j < BUFSIZE; j++)  {
      testTmp[j] = 'a';
    }
  }
   .....
  start = clock();
  for(i=1; i <= MAX; i++) {
    for(j=0; j < BUFSIZE; j++)  {
      buf[j] = testTmp[j];
    }
  }
  diff = diff + (float)((clock()-start)/((double)1000000));
  printf(" Total %4d th execution %.3f sec \n", MAX, diff);
  return 0;
}
```

그림 12. 경량 리눅스 운영체제 성능 측정을 위한 테스트 프로그램
Fig. 12. Performance Test Program for Lightweight O.S

그림 12.는 경량 리눅스 운영체제 성능 측정을 위한 테스트 프로그램이다. 이 프로그램은 메모리의 버프 공간 내에 일정한 값을 넣고, 그 값들을 계속하여 복사, 이동, 삭제를 한다. 이 작업은 4개의 for문에 의해 이루어진다. 그 시간을 각각의 for문 별로 측정한 후 몇 차례 반복 작업으로 나온 시간을 기준으로 측정한다. 이 프로그램의 성능 측정 결과는 아래에 보여준다.

위 그림 12.에서와 같이 4개의 for문을 각각 수행하게 한 후 그 값을 누적 적용시켜 최종 수행 시간을 구하는 형식으로 실험하였다.

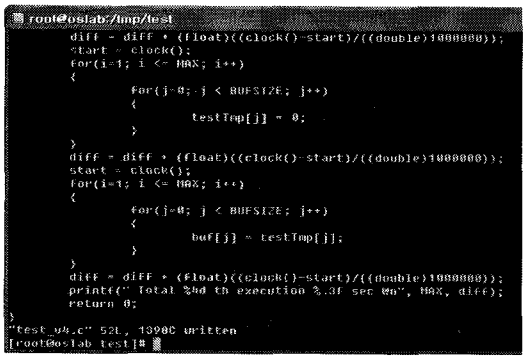

그림 13. 성능 측정을 위한 프로그램 소스 코드 화면
Fig. 13. Source Code for Performance Test Program

그림 13.은 경량 리눅스 운영체제의 성능 측정을 위하여 4개의 for문을 각각 수행한 결과를 얻어내는 프로그램을 작성하는 과정이다. 이 프로그램의 테스트 화면은 아래 그림 14., 그림 15.와 같이 나왔다. 본 테스트 프로그램을 수행하기 위하여 E-IDE 시스템과 DOM 메모리 내에 리눅스 운영체제를 설치하였다. 동일한 두 시스템에서 성능 측정 결과는 다음 그림 14., 그림 15.와 같다. 그림 14는 E-IDE 시스템에서 성능 측정 결과이다. 성능 측정 결과가 4664초 나왔다. 그림 15.는 DOM 메모리 내의 경량 리눅스 운영체제에서 성능 측정 결과이다. 성능 측정 결과가 4634초 나왔다.

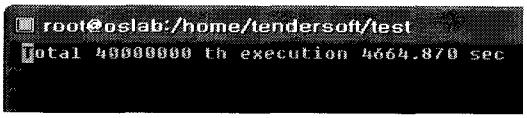

그림 14. E-IDE 장착 후 테스트 화면
Fig. 14. Test Screen for E-IDE System

그림 15. DOM Memory 장착 후 테스트 화면
Fig. 15. Test Screen for DOM Memory System

그림 14., 그림 15.에서와 같이 성능 측정 결과를 보면 일반 IDE 하드디스크 내의 리눅스 운영체제보다 DOM 메모리 내의 경량 리눅스 운영체체의 성능이 좋음을 알 수 있다. 표 2.는 E-IDE 시스템 내에서 리눅스 운영체제와 DOM 메모리 내에서 리눅스 운영체제에서 성능 측정 결과를 보여준다.

표 2. 두 시스템에서 성능 측정 결과
Table 2. Performance Comparison

실험방법	E-IDE (하드디스크)	DOM(Disk On Module)
BUFSIZE 기록	28.850	28.855
BUFSIZE 기록, 복사, 삭제	115.760	115.750
4번의 FOR문 추가	4664.870	4634.160

표 2.에서 보는바와 같이 성능 측정 프로그램에서 처음에는 단순히 BUFSIZE에 기록만 할 경우이다. 이 경우는 DOM(Disk On Module) 메모리 내 리눅스 운영체제의 성능이 조금 좋지 않음을 알 수 있다. 그 다음으로 버퍼에 기록, 복사, 삭제 작업을 반복하게 하였을 때에 두 시스템 간에 성능 차이가 나타남을 알 수 있다. 최종적으로 이 작업을 4번의 FOR문장을 이용하여 실험하였다. 본 논문에서 수행한 성능 측정결과에서 기존의 E-IDE 상의 리눅스 운영체제의 동작이 DOM 메모리내의 리눅스 운영체제 보다 하드웨어 적으로 빠르다. 그러나 본 논문에서 제안하는 DOM 메모리 내의 경량 리눅스 운영체제의 성능이 좋은 이유는 리눅스 운영체제가 경량화되어 동작하는데 소요시간이 적게 걸리기 때문이다.

V. 결 론

본 논문에서 제안한 경량 리눅스 운영체제의 설계 내용은 기존의 리눅스 운영체제에서 DOM 메모리 내에서 동작이 가능하도록 필수적인 모듈을 중심으로 커널 모

듈을 설계한다. 본 논문에서 설계한 경량 리눅스 운영체제의 모듈은 DOM 메모리 내에 상주하여 임베디드 시스템 전용 운영체제로 동작하게 된다. 본 논문에서 설계한 경량 리눅스 운영체제는 커널 내에서 기본적인 기능으로 프로세스 관리, 메모리 관리, 파일 시스템 관리, 입출력 관리 기능을 가지고 있다. 또한, 커널 기능을 이용하여 사용자가 시스템을 사용할 수 있도록 라이브러리와 명령어 등을 경량화시키는 작업을 수행한다. 이러한 설계 작업이 마무리되면 가상 파일 시스템을 구축하여 실제 시스템에 필요한 운영체제 이미지 구축 작업을 수행한다. 64KB 리눅스 이미지 구축을 위하여 "dd" 리눅스 명령어를 이용한다. 이러한 과정을 거치고 난후 경량 리눅스 운영체제 이미지 생성이 마무리되면 이 이미지를 이용하여 실제 DOM 메모리 내에 생성된 경량 리눅스 운영체제 이미지를 구동시킨다. DOM(Disk On Module)만으로 부팅과정을 할 수 있도록 Lilo를 설치한 후 부팅이 되면 이 작업은 마무리가 된다. 여기까지 작업은 일반 PC에 DOM(Disk On Module)을 사용할 수 있다. 본 논문에서 제안한 경량 리눅스 운영체제의 성능을 측정하기 위하여 성능 측정을 위한 벤치마크 프로그램을 작성한다. 작성한 프로그램을 이용한 제안한 경량 리눅스 운영체제 성능을 측정하였다. 실험 결과에서 DOM 메모리내의 리눅스 운영체제의 성능이 좋음을 알 수 있다.

참고문헌

[1] Linux Kernel Programming, ADDISON WESLEY, Beck, 2002.

[2] Korea Embedded Linux Project, http://www.kelp.or.kr

[3] 리눅스 커널 분석 2.4, 박장수, 가메출판사, 2003.

[4] Daniel P.Bovet, Marco Cesati, "Understanding the LINUX KERNEL", O'Reilly, 2001.

[5] Robert Love, 임베디드 개발자를 위한 리눅스 커널 심층 분석, Developer's Library, 2004.

[6] Karim Yaghmour, "Building Embedded Linux Systems", O'Reilly, 2003

[7] Skawratananond, Chakarat, "Unix to Linux Porting", Prentice Hall, Apr. 2006

[8] W. Richard Stevens, "Advanced Programming in the UNIX(R) Environment", Addison-Wesley, Jun. 1992

[9] W. Richard Stevens, "Unix Network Programming: The Sockets Networking API(Updated)", Addison-Wesley Professional, Nov. 2003

[10] W. Richard Stevens, "UNIX NETWORK PROGRAMMING VOLUME 2,2/E", Prentice Hall, Aug. 1998

[11] Maurice J. Bach, "The Design of the UNIX Operating System", Prentice Hall, Feb. 2000.

[12] 김영곤, "임베디드 시스템의 효율적인 리소스 관리를 위한 새로운 기법 개발", 한밭대학교 석사학위논문, 2006. 2.

[13] Eric S. Raymond, "Art of UNIX Programming", Addison-Wesley, Sep. 2003

[14] Syed Mansoor Sarwar, "Unix - A Textbook 2nd Edition", RADDISON WESLEY, Aug. 2004

[15] Preston, W. Curtis, "Unix Backup & Recovery", Oreilly & Associates Inc, Nov. 1999.

[16] Worldwide Embedded Operating Environments Forecast, 2003-2007, IDC #29308,2003,5.

[17] 박길성, "임베디드 리눅스 시스템을 이용한 홈네트워크 제어에 관한 연구", 순천향대학교 석사학위논문. 2005.

저자소개

장 승 주(Seung-Ju Jang)

1985년 부산대학교 계산통계학 (전산학) 학사
1991년 부산대학교 계산통계학 (전산학) 석사
1996년 부산대학교 컴퓨터공학과 박사
1987년~1996년 한국전자통신연구원 시스템 S/W연구실
1993년~1996년 부산대학교 시간강사
2000년~2002년 University of Missouri at Kansas City, visiting professor
1996년~현재 동의대학교 컴퓨터공학과 교수
※관심분야 : 운영체제, 임베디드 시스템 운영체제, 분산 시스템, 시스템 보안