uClinux 기반의 멀티카드리더기 모듈 인터페이스 최적화 방안에 관한 연구 및 구현

하성준° 김홍규 문승진
수원대학교 IT대학 컴퓨터학과
{e11even21°, exxxfx, sjmoon}@suwon.ac.kr

A Study and Implementation of a multi cardreader module interface optimizer based on the uClinux

Sung-jun Ha° Hong-Kyu Kim Seung-jun Moon
Dept. of Computer Science, The University of Suwon

요 약

멀티미디어 디지털 기기의 발전으로 다양한 형태의 저장장치에 저장되어 있는 디지털 데이터 등을 다른 멀티미디어 디지털 기기에서 이용하기 위한 외부 인터페이스로 멀티카드리더기를 사용하며 이러한 멀티미디어 디지털 기기는 임베디드 리눅스를 기반으로 하고 있다. 이에 본 논문에서는 멀티미디어 디지털 기기에서 사용되는 uClinux기반의 운영체제와 멀티카드리더기 모듈과의 인터페이스 최적화 방안에 관하여 제안한다. 기존의 멀티카드리더기 모듈과의 인터페이스 방법은 시스템의 리소스를 많이 사용하여 시스템의 안전성이 떨어지며 성능 또한 좋지 못하다. 이러한 문제를 디바이스 드라이버와 운영체제 인터페이스를 최적화 하는 방법을 본 논문에서 논의 하고자 한다.

1. 서 론

멀티미디어 디지털 기기의 발전은 디지털 데이터의 활용과 저장 장치의 발전을 도모하였다. 디지털 기기의 저장 장치로 활용되는 메모리 카드는 디지털 기기의 제조사와 특성에 따라서 여러 종류로 구분 된다. 여러 종류의 메모리 카드를 사용하는 사용자입장에서 다양한 메모리 카드를 한번에 인식할 수 있는 카드리더기를 필요로 하게 되었으며 멀티 카드리더기가 보급 되었다. 멀티 카드리더기를 활용하여 하나의 장치에서 다양한 종류의 메모리 카드를 인식하여 작업 할 수 있으며, 최근에는 임베디드 장치에 내장형 멀티 카드리더기가 부착되어 사용되고 있다.

임베디드 장치의 내장형 운영체제는 대부분 임베디드 리눅스를 사용하고 있으며 일부에서 포켓 PC또는 WinCE를 사용하고 있으나 포켓 PC나 WinCE는 다양한 디바이스 드라이버와 커널간 모듈 인터페이스가 상당히 잘 구성되어 있다. 하지만 대부분의 임베디드 장치에서 사용되고 있는 오픈 소스 기반의 임베디드 리눅스기반에서의 멀티카드리더기 모듈은 커널과의 인터페이스 문제점을 갖고 있다. 이러한 문제점은 임베디드 장치의 제약조건으로 인한 멀티 카드리더기의 모듈과 운영체제 커널의 인터페이스를 잘못 구현하여 발생하며 시스템 리소스의 불필요한 낭비로 인하여 다운되는 안정성의 문제가 발생한다.

이에 본 논문에서는 임베디드 리눅스 기반의 uClinux를 사용하여 멀티 카드리더기와 운영체제 커널간의 안정성 문제를 해결하기 위한 멀티 카드리더기 디바이스 드라이버와 운영체제 커널 인터페이스를 최적화 하는 방법에 관하여 제안하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서 멀티 카드리더기와 uClinux에 관하여 간략하게 설명하였고, 3장에서 멀티카드리더기 사용을 위한 디바이스 드라이버 모듈과 운영체제 인터페이스 설계 및 구현에 관하여 논의하고 4장에서는 본 논문의 결론과 앞으로 해결해야 할 과제들을 제시한다.

2. 관련연구

2.1 Card Reader

멀티 카드리더기는 디지털카메라나 PDA 등에 사용되는 SD 카드 또는 CF카드 Sony 메모리 스틱 등과 같은 다양한 종류의 메모리 카드를 슬롯에 장착해 USB 인터페이스로 PC등과 연결해 데이터를 송수신 할 수 있다. 이러한 멀티 카드리더기를 사용하면 기기와 PC간 연결해야 하는 케이블이 필요 없으며, USB 단자와 연결하면 별도 드라이버 설치 없이 바로 외장형저장장치로 인식하기 때문에 사용과 설치가 간편하다는 장점이 있다.

멀티 카드리더기는 고급형 완제품 데스크톱PC에 장착됐지만, 올해 들어 보급형 데스크톱PC 뿐 아니라, 노트북에도 자체 내장되는 추세다. 프린터도 기본 장착한 제품이 등장해 메모리카드에 있는 디지털이미지를 바로 출력할 수 있으며, 프로젝터 자체에 리더기를 내장해 PC연결 없이 메모리카드 만으로 프리젠테이션이 가능케 하는 등 활용 범위가 갈수록 넓어지고 있다.

한 자료에 따르면 대만과 중국 등 제조업체들은 카드리더기 시장이 60%이상 성장할 것으로 예상하고 다기능 멀티미디어 개발이 한창이다. 일부 업체는 20여 가지 포맷을 지원하는 제품을 개발하고 있으며, 하드디스크드라이브를 내장해 PC없이 데이터를 저장하고, 제품에 장착된 자체 LCD화면을 통해 멀티미디어 파일을 재생할 수 있는 제품도 준비중이며 앞으로 멀티리더기는 PC의 기본품목이 될 것이며, 제품의 크기를 대폭 줄이거나, 메모리 내장 또는 멀티미디어 엔터테인먼트 기능을 결합시킨 제품 등 다양한 형태로 성장할 것으로 전망된다.

2.2 uClinux

Mobile Device에대한 사람들의 관심이 높아지면서, 기존의 Embedded 장비에 linux를 적용하고자 하는 사람들이 많아지고 있다. 게다가 linux가 네트워크 분야에서 많은 장점을 가지고 있고, 여타의 embedded OS에 비해 개발 환경 구축 및 개발 작업이 손 쉽게 이루어 지면서, 저렴한 비용으로 사용이 가능하므로, embedded linux에 대한 관심도는 가히 폭발적으로 증가하고 있다. 일반적으로 사람들이 사용하고 있는 서버용 내지는 데스크 탑 용의 리눅스 배포판과 같이, Embedded Linux에도 여러가지 배포판이 존재한다. 그 중에서, MMU-less Processor를 위한 embedded linux의 한 종류인 uClinux 임베디드 리눅스 분야에서 널리 알려져 있는 리눅스 커널의 변형된 모습 중 하나이다.

uClinux (MicroController Linux)는 1998년에 Dragonball 68k 시리즈에 처음으로 포팅 되었으며, 그 이후로 수많은 타겟 프로세서에 포팅이 되었다. 이는 uClinux는 MMU(Memory Management Unit)를 가지고 있지 않은 32비트 프로세서를 지원하기 위하여 MMU지원 기능을 커널에서 삭제하였기 때문에, 현재 데스크 탑 또는 서버에서 널리 사용되고 있는 주류의 Linux 커널과는 구별되는 리눅스 커널이다. 또한 uClinux는 작은 자원을 가지고 있는 마이크로 컨트롤러에서 사용하기 위해, footprint가 매우 작으며, BFLT(Binary Flat) 실행 형태를 가지고 있다.

3. Using a USB Card Reader in uClinux

3.1 Device Driver Porting

대부분의 임베디드 리눅스기반의 플랫폼은 카드리더기를 포함한 모든 USB 드라이버를 SCSI디바이스로 인식되므로 SCSI드라이버를 설치해주어야 한다. 따라서 커널 컴파일의 "SCSI Support" 부분의 옵션을 설정하며, 카드리더기에 접근하기 위해서 "USB Support"부분의 옵션도 설정 해주어야한다. 환경설정 및 모듈의 목록은 표 1과 같다.

표 1 configuration name and the module name

<SCSI Support>

- SCSI Support (CONFIG_SCSI, scsi.o)
- SCSI disk support (CONFIG_BLK_DEV_SD, sd_mod.o)
- SCSI generic support (CONFIG_CHR_DEV_SG, sg.o)
- Probe all LUNs on each SCSI device (CONFIG_SCSI_MULTI_LUN)

<USB Support>

- Support for USB (CONFIG_USB, usb.o)
- Preliminary USB device file system (CONFIG_USB_DEVICEFS)
- EHCI HCD (CONFIG_USB_EHCI_HCD, usb-ehci-hcd.o),
- UHCI (CONFIG_USB_UHCI, usb-uhci.o),
- OHCI (CONFIG_USB_OHCI, usb-ohci.o)
- USB Mass Storage support (CONFIG_USB_STORAGE, usb-storage.o)

<SCSI Support>

- *SCSI Support* : SCSI 하드 디스크나, 테이프 드라이브, 시디롬, 혹은 그 밖의 다른 SCSI 장비 지원.
- *SCSI generic support* : CD-Writer, 스캐너, 신디사이저 등 장치의 사용.
- *Probe all LUNs on each SCSI device* : 하나 이상의 LUN(논리장치번호)을 지원하는 SCSI장치가 장치되어 있는데도 단일 LUN만이 인식될 때 강제로 SCSI 드라이버가 여러 LUN을 검색. 멀티 카드리더기를 사용을 위해서는 필수.

<USB Support>

- *Support for USB* : Universal Serial Bus (USB)의 사용.
- *Preliminary USB* device filesystem :"/proc file system support" 옵션과 이 옵션을 선택하면 /proc/usb/devices에서 USB 버스에 연결된 장치를 알 수 있다.
  /proc/usb/drivers 에서는 현재 로드된 USB 커널 클라이언트 드라이버를 확인할 수 있다.
- *UHCI Alternate Driver (JE) support* : Universal Host Controller Interface 는 피시에서의 USB 하드웨어 접근에 대한 Intel 사의 표준이다.
- *OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support* : Open Host Controller Interface 는 Compaq/Microsoft/National 의 USB 표준이다. 대부분의 비 인텔 기종과 인텔 칩셋을 사용하지 않는 일부 x86 호환 기종.

3.2 Multi Card Readers

멀티 카드리더기의 사용을 위해서는 *"Probe all LUNs on each SCSI device(CONFIG_SCSI_MULTI_LUN)"* 옵션을 커널에서 사용 할 수 있어야 한다. 만약 옵션에 없다면 */etc/modules.conf* 부분에 *options scsi_mod max_scsi_luns=8* 을 추가 해주어야 한다. */proc/scsi/usb-storage-0/*을 확인하거나 카드리더기를 연결하는 순간 Hot Plug에 의해 확인할 수 있을 것이다.

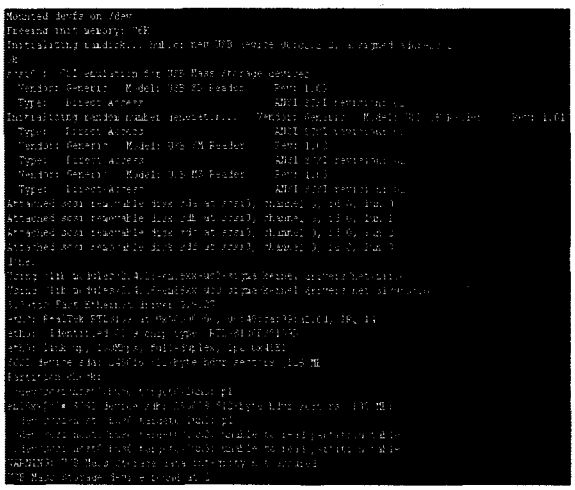

그림 1 멀티 카드리더기 인식

- 현재 lun0과 lun1의 볼륨(4개의 멀티 카드리더기 슬롯 중 2개의 슬롯)에 메모리 카드가 들어가 있으며 용량 및 경로에 대한 설명이 나온다. 메모리 카드가 삽입유무에 따라 디렉토리 내의 차이가 생긴다.

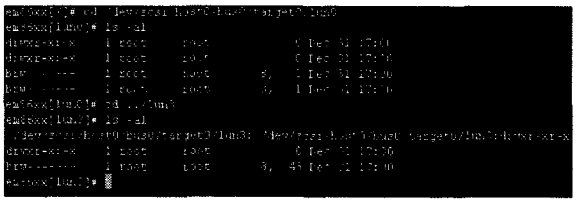

그림 2 메모리카드의 삽입시 변화

- 메모리카드가 삽입된 슬롯(lun0)과 미 삽입된 슬롯(lun3)의 차이는 ***part1*** 파일의 유무이며 이 파일을 원하는 경로에 마운트 시켜야 메모리카드를 사용 할 수 있다

3.3 Multi Cardreader Module Interface 구현

멀티 카드리더기를 커널에서 인식을 하였다 해도 그 상태로 바로 접근하여 사용할 수 있는 것이 아니다. 'mount' 명령어가 필요한데 멀티 카드리더기의 경우에는 약간의 복잡한 면이 있어 초보자에게는 쉽지 않을 것이다. 멀티 카드리더기의 경우엔 lun 볼륨이 여러 개 이므로 자신이 사용하고자 하는 카드, 즉 lun 볼륨을 선택하여 ***'옵션 -t vfat'*** 파일시스템 타입을 입력 해주어야 비로소 사용 할 수 있는 것이다.

표 2 멀티 카드리더기 모듈 간 인터페이스

```
void *usbCheck(void *p)
{
  while(1){
  if ( (USB_Attached==0) && (usb_umount==0) ){
          context.gui->GetUsb()->StartUsbMount();
          usb_umount =
context.gui->GetUsb()->UsbStateCheck();

  }
  else if ( (USB_Attached==0) && (usb_umount==1) )
  {
USB_Attached = 1;
   context.gui->GetUsb()->StartUsbMount();
   usb_umount = context.gui->GetUsb()->UsbStateCheck();
    context.gui->Set_m_enableUSB(TRUE);
    if(context.player != 0){
      state = context.gui->get_m_playMode();
      FileType = sambaGui.get_FileType();
      if   (FileType   ==   FILETYPE_VIDEO||FileType   ==
FILETYPE_PICTURE){
        if ( state == STATE_CLOSE || state == STATE_STOP
){
          context.gui->RestoreOsd();
        }
      } else {
          context.gui->RestoreOsd();
      }
    }
  } else if ( (USB_Attached==1) && (usb_umount==0) ) {

    context.gui->Set_m_enableUSB(FALSE);
    state = context.gui->get_m_playMode();
    FileType = sambaGui.get_FileType();

    if   (FileType   ==   FILETYPE_VIDEO||FileType   ==
FILETYPE_PICTURE){
      if ( state == STATE_CLOSE || state == STATE_STOP ){
        context.gui->RestoreOsd();
      }
    } else {
      context.gui->RestoreOsd();
    }
    if (sambaGui.get_usbpath() == 1)
    {
      context.gui->SendKey(RM_HW_STOP);
if (context.gui->Get_m_enableHarddisk())
        {
          context.gui->SendKey(RM_HW_HDD_MOUNT);
        } else if (context.gui->Get_m_enableNetwork()) {
          context.gui->SendKey(RM_HW_NET_MOUNT);
        } else {
          context.gui->SendKey(RM_HW_START);
        }
    }
   context.gui->GetUsb()->StopUsbMount();
          usb_umount =
context.gui->GetUsb()->UsbStateCheck();
    USB_Attached = 0;
  }
  else if ( (USB_Attached==1) && (usb_umount==1) ) {
          context.gui->GetUsb()->StartUsbMount();
          usb_umount =
context.gui->GetUsb()->UsbStateCheck();
          }

 }
}
```

- GetUSb()에 의해 메모리카드의 유무가 인식되며

USB_Mount(), USB_Umount()에 의해 삽입 시 자동으로 mount, 제거 시 umount가 되어 사용자가 편리하게 사용할 수 있다.이러한 방식에서 속도가 현저히 느려지는 오류가 발생하였다.이 오류는 Thread 처리로 해결이 가능하다.

표 2에서와 같이 uClinux기반의 임베디드 시스템에서의 사용자 인터페이스는 주로 C와 C++를 활용, GUI 라이브러리를 참조하여 설계된다. 임베디드 시스템에서의 UI 수정 작업은 소스코드를 수정한 후 컴파일하고 임베디드 장비에 다시 포팅 하는 과정을 거치게 되는데, 이때 비교적 많은 시간이 걸리게 된다. 또한 소스코드 설계구조에 대해 자세히 알고 있지 못하다면 수정하는데 많은 시간이 걸리게 마련이다.

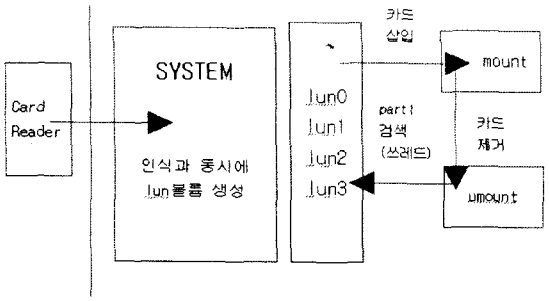

그림 3 CardRead Module에 의한 Interface 구현 및 Thread 처리

· Thread : RMCreateThread("UsbThread", usbCheck, NULL); 속도 저하 및 멀티태스킹 가능.

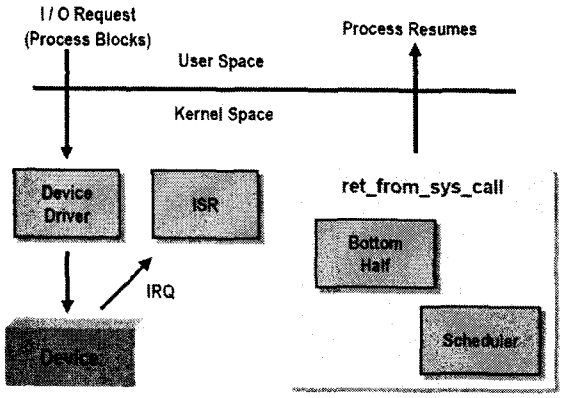

그림 4 디바이스 입출력 요청 - 인터럽트(Interrupt)

· I/O 작업 수행 시 CPU는 다른 작업을 수행하고, 실행완료와 같은 비동기적인 사건이 발생했을 때 디바이스가 CPU에게 알려주는 방식이다.
· 인터럽트가 발생하면 CPU는 원래 작업의 문맥을 저장하고, 인터럽트 처리.

그림 3과 그림 4에서와 같이 많은 하드웨어들은 프로세서의 사용 효율을 높이기 위해서 인터럽트와 조합하여 입출력 처리를 수행한다. 즉 입력과 출력 처리에 대한 동기를 맞추도록 디바이스 드라이버를 작성한다. 이런 처리를 위해서는 디바이스 드라이버에 입력 조건이나 출력 완료 조건이 발생할 때까지 프로세서를 재우는 루틴은 필수적이다. 저수준 파일 입출력 함수를 이용하여 이런 디바이스 드라이버와 연관된 디바이스 파일에 접근하면 프로세스의 수행이 정지되는 현상이 발생한다. 이런 식으로 처리하는 것은 여러 디바이스 파일에서 데이터를 읽어오거나 데이터를 써넣을 경우에 매우 유용할 수 있다.

4. 결론

본 논문에서는 멀티 카드리더기 디바이스 모듈과 임베디드 운영체제의 하나인 uClinux의 인터페이스를 최적화 하여 시스템 리소스의 불필요한 사용을 최소화 하여 임베디드 장비의 안정성을 향상시키기 위한 방법에 관하여 제안하였다. 제안된 방법은 멀티 카드리더기의 디바이스 모듈과 임베디드 운영체제의 인터페이스를 하나의 모듈로 구성하고, 멀티 카드리더기의 카드 리딩 방식을 기존의 *Polling*방식에서 *Interrupt*호출 방식으로 재구성하여 Polling방식의 불필요한 시스템 리소스 사용을 최소화 하였다. 제안된 방식의 Interrupt 호출 방식을 사용함으로서 시스템 리소스의 최적 사용과 함께 안정성이 보다 향상되었다. 이러한 방법을 경량 임베디드 시스템에 사용하게 된다면, 적은 시스템 리소스로 멀티 카드리더기의 인터페이스를 최소화 하여 적재 할 수 있다. 향후 본 논문을 바탕으로 소형화된 임베디드 시스템에 적용하여 발생될 수 있는 멀티 카드리더기 모듈과 uClinux 만이 아닌 이기종의 임베디드 운영체제와의 문제점을 해결할 수 있도록 연구할 것이다.

참 고 문 헌

[1] Simon Fraser University, Computer Science (http://www.cs.sfu.ca/)
[2] 유영창, "리눅스 디바이스 드라이버", 한빛미디어, 2004
[3] "Korea Embedded System for Linux", (http://www.kesl.co.kr)
[4] "Korea Embedded Linux Project", (http://www.kelp.or.kr)