논문 2018-1-4

임베디드 기기 감정에서 디바이스드라이버 유사성 설정

이규대*

A Similarity of device driver on Embedded system

Kyu-Tae Lee*

요 약

저작물 감정은 원 개발자의 저작물과 복제로 의심되는 저작물과의 유사성을 판별하는 작업이다. 공정하고 객관적인 검증을 위해서는 개발자와 피개발자의 목적물이 제출되어야하나 대부분의 경우 피의자측의 불성실한 자료제출로 일대일 비교가 어려운 상황이 발생한다. 특히 임베디드 시스템의 경우, Linux OS의 kernel을 탑재하고, 그에 따른 응용프로그램이 디바이스드라이버를 이용하여 실행되도록 제작된다. 이러한 시스템이 복제 또는 도용되는 경우, 기존과 같은 비교방법으로는 감정결과를 얻을 수 없다. 운영방식의 차이, 그리고 사용언어에 따른 프로그램의 구조적 차이 등이 분석을 어렵게 한다. 또한 운영체제의 경우 공개 프로그램 비중이 높아 유사성 비교 결과가 다르게 나타난다. 본 논문에서는 두 제품의 디바이스 드라이버에 대한 비교 방안을 제시하였다.

Abstract

Copyright evaluation is the work of judging the similarity between the product of the original developer and works suspected of duplication. In order to keep the fair and objective verification, the developer and the developer's object should be submitted, but in most cases, it is difficult to compare one-on-one with the unfair disclosure of data by the suspect. In particular, in the case of an embedded system, a kernel of the Linux OS is mounted and an application program is made to be executed using the device driver. When such a system is duplicated, evaluation results can not be obtained by the conventional comparison method, the difference of the operating system, and the structural difference of the program according to the nguage used make analysis difficult. By this reason, the similarity comparison result is different because of the high proportion of open programs. In this paper, we compare and analyze the kernel source and device driver of two products.

한글키워드 : 커널 감정, 디바이스드라이버, linux 유사성, 하드웨어감정 항목

keywords : kernel evaluation, device driver, linux similarity, hardware sub-items

1. 서 론

정보기기는 프로세서를 기반으로 하는 입출력 하드웨어와 운영체제를 기반으로 하는 응용소프트웨어의 일체형으로 구성된다. 시스템의 제작자는 독자적인 기능의 구현과 성능개선을 위한 목적으로 특정한 프로세서와 특정 입출력 인터페

* 공주대학교 정보통신공학부
(email: ktlee@kongju.ac.kr)
접수일자: 2018.05.23. 심사완료: 2018.06.10.
게재확정: 2018.06.20.

이스를 갖는 하드웨어를 설계하고 제품으로 구현하는데, 임베디드 시스템의 구성으로 제작되는 경우에는 운영체제를 사용한 응용기기를 제작하게 되며, 이 경우 많이 사용되는 OS가 Linux Kernel 이다[1]. 그러나 리눅스커널은 압축파일 형태로 시스템에 탑재되어, 소스의 내용을 판독하기 어렵게 구성된다.

정보기기 분쟁에서 소스코드 기준의 감정을 어렵게 하는 부분이 프로그램 소스의 불성실한 제공이다. 감정인이 정확한 유사성 도출을 위해서는 분쟁 당사자들의 자료제공이 선행되어야 하나, 복제 의심을 받는 측에서 소스코드를 거부하는 경우, 제품의 내부에 저장된 실행코드만으로 검증해야 하는 상황들이 발생한다.

본 연구는 커널코드에 대한 양측의 소스코드가 제공된 상황을 가정하고, 디바이스드라이버의 유사성비교 과정 및 유사도의 객관성 확보에 대해 다룬다.

2. IT 정보기기 하드웨어 구성

정보기기는 휴대가능한 정도의 크기로 제작되는 경우, 전원사용의 시간과 내부 프로세서의 처리용량의 제한으로 기능의 제한이 따르는 단점이 있다. 이러한 시스템의 내부구성을 보면 그림1과 같이 프로세서를 중심으로 프로그램의 임시저장소로 활용하는 RAM, 응용프로그램을 저장하는 FLASH ROM, 정보기기의 입출력용으로 사용되는 키보드, 디스플레이장치 및 터치패드, 그리고 외부 통신이 가능하도록 하는 통신포트(TCP/IP, USB, COM) 등이 연결된다.

설계된 하드웨어적인 구성을 바탕으로 개발자는 익숙한 프로그램언어를 사용하여, 응용프로그램을 제작하고, 번역하고, 완성된 프로그램 실행코드를 FlashROM에 저장하여 완성한다. 이때 응용프로그램은 프로세서의 주요 인터페이스 기능과 각종 주변장치(key pad, LCD monitor, LCD touch screen, 통신포트) 를 효과적으로 활용하여, 사용자에게 유용한 기능의 단말기기를 제작한다.

작성한 프로그램은 compile 과정을 통해 특정 프로세서에 일치하는 실행코드로 완성된다. 하드웨어는 design 작업을 통해 그림2와 같이 사용하기 용이한 제품으로 완성된다.

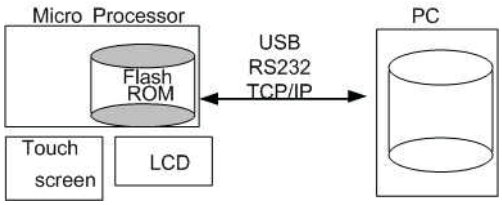

그림 1. 마이크로 컨트롤러 인터페이스 구성
Fig. 1. Micro controller interface

단말기의 기능이 다양해지고, 핵심프로세서의 기능이 고기능화 됨에 따라서, 임베디드 구조로 개발된 정보기기는 그림2와 같이 운영체제(OS)로 알려진 시스템관리용 운영프로그램이 탑재되어 제작된다. 운영체제는 일반 컴퓨터의 OS와 유사한 기능으로 주변기기, 메모리, 파일, 인터페이스 등 시스템의 모든 기능을 관리하는 능력을 가진 소프트웨어를 의미한다.

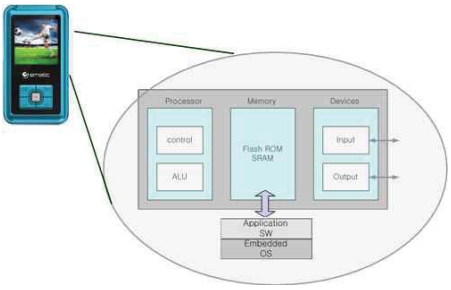

그림 2. OS 기반 정보기기 구조
Fig. 2. Device configuration with OS

이러한 OS가 탑재되어, 단말기를 실시간으로 동작시키는 기기를 임베디드 정보기기라 하고, 소프트웨어 부분은 특정기능의 응용프로그램에 운영체제가 추가되어, 실행 가능한 프로그램으로 제작된다.

3. 임베디드운영체제 구성

정보단말기는 사용자의 요구에 의해 새로운 기능이 계속 추가되고 있으며, 특히 동일한 입력장치에서 다양한 기능을 구분하기 위해 시퀀스 지연기능을 사용하고 있다. 이 방법은 하드웨어적인 기술 및 소프트웨어적 기술이 연결된 기능으로, 유사도 감정시 응용 프로그램이나 하드웨어 모듈만의 비교방법에 포함되어야 할 항목이다.

정보기기의 하드웨어는 그림 3과 같이 프로세서를 중심으로 하는 입출력부분, 프로그램저장부분, 통신부분으로 구분할 수 있다. 최근에 출시되는 프로세서는 자체적으로 다양한 인터페이스 블록을 내장하고 있기 때문에 프로세서가 입출력 기능을 포함하는 경우가 많아지고 있다. 정보기기 하드웨어의 기능에 따른 유사성 비교는 표준화된 방식에 의한 부분을 제외하고, 응용프로그램에서 작성된 프로그램을 비교하고, 특정 인터페이스 칩이 사용되는 경우를 고려하여 판단하는 것이 타당하다.

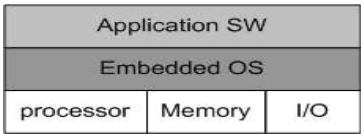

그림 3. 임베디드 시스템 구조
Fig. 3. Embedded system architecture

응용프로그램은 사용자인터페이스부분으로 사용자에게 편리한 입력기능과 출력기능을 제공하여, 정보기기 사용자의 기기활용도를 높이는 부분이다. 사용목적에 따라 기기의 하드웨어 부분과 접속되며, 운영체제의 기능을 효과적으로 사용한다.

정보기기의 기능은 이 응용프로그램의 동작에 의해 결정되는 것으로 제작자는 사용자의 요구에 적합한 응용프로그램은 많이 제작해서 보급하고 있다. 따라서 응용프로그램은 제작자 고유의 기능과 아이디어가 포함된 부분으로 응용프로그램의 기능비교를 통해 기기의 유사성 여부를 도출할 수 있다.

3.1 시스템의 설치 환경

임베디드 시스템이란 특정 프로세서의 메모리에 운영체제(OS)와 운영프로그램(kernel)을 장착하고, 이를 이용하여 특정한 입출력장치(LCD, button)로 동작되는 시스템을 말한다.

이러한 시스템을 구성하기 위해서는 동작에 적당한 프로세서를 선정하고, 그 프로세서에 설치 가능한 운영체제(OS)를 선정한 후, 적정한 방법으로 프로세서의 메모리에 실장을 시키고 동작이 가능하도록 설치하여야 한다.

초기 아무것도 설치되어있지 않은 시스템에 OS와 kernel을 설치하는 과정을 알아보고, 이러한 시스템을 운영하기위한 절차 및 개발방법을 기술한다.

3.2 임베디드시스템의 설치

임베디드 시스템은 사용하는 프로세서와 운영체제에 따라 설치방법이 상이하기 때문에 여기서는 EMPOS-tiny 보드를 중심으로 설치과정을 알아본다. 이 보드는 intel Xscale PXA255 프로세서를 사용하고 있으며, 유용한 I/O장치(LCD, button, IrDA, GPIO, Ethernet 등)를 가지고 있다. 초기 아무것도 설치되어있지 않은 시스템에

운영체제와 사용자프로그램을 설치하기 위해서는 시스템에 전원이 들어왔을 때 동작이 가능한 부트로더라는 프로그램을 설치해 주어야 하고, 이 부트로더에 의해 운영체제(OS)가 설치된다.

모든 프로세서는 초기 전원이 들어왔을 때 처음 시작하는 주소에서 프로그램이 시작되도록 만들어지기 때문에, 보통 이 시작주소에 부트로더가 있는 곳으로 이동하는 명령이 위치한다. 부트로더가 실행되면 몇가지 명령어에 의해 자체 실행을 하거나, 포팅할 OS를 메모리의 특정 위치에 저장하고 실행하는 역할을 담당한다.

또한 사용하는 운영체제 및 사용자 프로그램의 개발결과를 시스템에 이식하기위해서는 시스템과 PC와의 연결이 필요하며, 이를 위해 PC와 target 시스템 간에 모니터링을 위한 RS232 케이블, 고속전송을 위한 Ethernet 케이블, Flash Rom의 read/write를 위한 JTAG cable이 연결된다.

이러한 케이블은 시스템의 개발 및 업그레이드를 위한 것이며, 실제 사용을 할 때는 사용되지 않는다. 호스트 PC는 target 시스템의 개발에 사용되는 것으로, 필요한 유틸리티와 개발 툴(tool)이 설치되어야 한다. 이때 타겟 시스템의 OS와 같은 운영체제가 요구되므로, 리눅스 OS가 사용되는 경우는 PC에도 운영체제는 리눅스로 설치되어 있어야 한다.

4. 디바이스 드라이버 비교

디바이스 드라이버는 커널과 제어를 필요로 하는 하드웨어 사이의 계층(layer)이다. 이 디바이스 드라이버는 추상적 개념으로. 커널이 장치와 직접 통신하지 않고 정의된 인터페이스로 각 디바이스 드라이버에게 맡기는 방식이다. 디바이스 드라이버는 문자 디바이스 드라이버, 블록 디바이스 드라이버, 네트웍 디바이스 드라이버의 3가지로 분류한다. 디바이스 드라이버들은 커널에 정적으로 링크 되어서 사용되기도 하며, 때로는 동적으로 운영체제가 실행중인 상태에서 커널과 링크 되거나 제거 될 수 있으며. 이를 위해 리눅스에서는 모듈기능을 제공하여 디바이스의 동적 적재, 제거를 가능하게 한다.

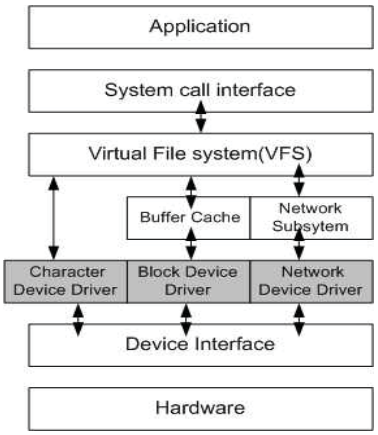

그림 4. 디바이스 드라이버 계층
Fig. 4. Device driver level

그림4와 같이 어플리케이션에서는 시스템 콜을 통하여 VFS(Virtual File System) 안의 장치 디바이스를 읽어 디바이스 드라이버에게 명령을 내리면 디바이스 드라이버는 해당되는 함수를 호출하여 하드웨어를 제어 한다 즉 디바이스 드라이버는 커널 레벨 프로그램이기 때문에 일반적인 사용자 어플리케이션을 작성할 때와 다른 규칙과 형태를 갖는다.

- 커널은 모드를 전환할 때 프로세스의 실수형 변수로 저장하지 않는다.
- 커널 레벨 프로그램에서는 CPU를 완전히 독점할 수 있으므로 다른 작업이 방해되지 않도록 작성한다.
- 커널 공간에서의 디버깅은 훨씬 더 어렵고, 드라이버는 하드웨어를 다룰 때 정교한 타이밍을 종종 필요로 하기 때문에 가능한 간결

한 코드로 작성한다.

디바이스 드라이버 함수의 작성은 헤더를 정의 하고, 사용될 함수들을 작성한다. 기본적으로 어플리케이션에서 open하고 close할 때 대응하여 동작하는 open과 release함수를 작성하고 디바이스 특징에 따라 file_operations의 구조체에 나와 있는 함수를 작성한다. 다음으로 작성된 함수들의 특성에 맞게 file_operations 구조체에 등록하고 init_module함수와 cleanup_module함수를 이용하여 디바이스 드라이버를 커널에 삽입하고 제거할 때 사용되는 함수를 구현한다. 리눅스 에서 모든 장치 디바이스는 하나의 파일처럼 동작하므로 각, 함수들의 file_operations구조체가 등록된다.

작성된 함수는 다음 그림5와 같다.

```
// Device driver sample.c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
int init_module(void)
{
printf("Hello, Device driver!\n");
return 0;
}
void cleanup_module(void)
{
printf("End, Driver !\n");
}
```

그림 5. 디바이스 드라이버 소스코드
Fig. 5. Device driver source code

작성된 소스코드(sample.c)컴파일 과정을 통해 실행파일(sample.o)로 작성된다. 커널의 삽입은 실행코드 파일로 작업되고, 해당 드라이버는 insmod 와 rmmod 를 사용하여 커널에 디바이스 드라이버를 삽입하고, 해제된다.

디바이스 드라이버의 특징은 하드웨어와 연결되는 위치(level)에 있기 때문에 그림6과 같이 LED가 하드웨어적으로 연결된 경우, 물리적인 주소(address)가 지정되어야 한다. 그림에서와 같이 지정된 포트의주소가 (LEDIOPORT_ADDRESS 0xf1600000) 동일한 값으로 표기된 소스코드가 발견된 경우는 디바이스 드라이버와 연관되는 하드웨어도 복제 도용의 가능성이 의심되는 상황으로 정밀 점검해야하는 근거로 도출된다.

```
// Device driver ledport.c
#include <linux/ioport.h>
#include <asm/uaccess.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <asm/io.h>
#define LEDIOPORT_MAJOR 0
#define LEDIOPORT_NAME "LED IO PORT"
#define LEDIOPORT_MODULE_VERSION "LED IO PORT V0.1"
#define LEDIOPORT_ADDRESS 0xf1600000
#define LEDIOPORT_ADDRESS_RANGE 1
//Global variable
static int ledioport_usage = 0;
static int ledioport_major = 0;
// define functions...
int ledioport_open(struct inode *minode, struct file *mfile) {
if(ledioport_usage != 0) return -EBUSY;
70 HBE-EMPOS-Tiny Software Manual
MOD_INC_USE_COUNT;
ledioport_usage = 1;
return 0;
}
```

그림 6. 디바이스 드라이버 led.c 소스코드
Fig. 6. Device driver led.c source code

커널소스의 비교는 공개프로그램이 차지하는 비중이 높아, 전체 커널을 대상으로 하는 비교는 유사도 결과의 객관성에 논란의 여지가 있다. 따라서 디바이스 드라이버, 라이브러리, 유틸리티 등의 세부프로그램을 대상으로 하는 유사성 비교가 요구된다.

이상과 같이 디바이스 드라이버는 소스코드가 확보되는 경우, 일반적인 방법으로 라인단위 유사도 검증을 통해 드라이버의 코드 유사성을 도출할 수 있다. 그러나 실행파일(sample.o)로 제공

되거나, 커널의 이미지로 제공되는 경우는 일반 감정의 실행화일 감정으로 수행되고, 감정의 결과에 정확성이 감소된다.

5. 결 론

리눅스 커널은 운영체제와 연결된 하드웨어 인터페이스 구성으로 디바이스 드라이버 프로그램으로 링크시키는 방법으로 구현된다. 따라서 커널의 비교는 개발환경에서 사용된 프로그램 소스가 제공되어야 비교가 가능하다, 그러나 많은 경우 개발소스를 제공하지 않는 사례가 있고, 이 경우 zImage 파일을 확보하여 간접적인 비교가 수행된다.

본 연구에서는 디바이스 드라이버의 작성과정을 예시하여, 유사성 도출 과정을 점검하였다. 커널소스의 전체비교는 공개프로그램의 비중이 높아 개발자의 소스코드내용이 유의성을 갖지못하는 것으로 확인되었다. 따라서 커널소스의 유사성 비교시에는 커널소스 전체비교보다는 라이브러리, 디바이스, 유틸리티 소스 등으로 세분화하여야 원저작권자의 독창적 내용을 판단할 수 있으며, 감정목적물 확보시 소스코드의 검증이 우성되어야 한다.

참 고 문 헌

[1] Ian Maxwell, Information_ Display_article.pdf Information display Dec. 2007.

[2] 이규대, “유사성 비교에서 세부항목 설정 기준”, 한국소프트웨어감정평가학회 논문지, 12권1호, pp.21-26, June, 2016.

[3] 임경수, 박종혁, 이상진, “디지털포렌식 현황과 대응방안”, 보안공학연구논문지, Nov. 2008.

[4] 류희수, “정보보호: 디지털 세상의 CSI, 그 가능성은”, 정보통신진흥협회, 2007.

[5] 조용현, “디지털 포렌식을 위한 절차와 도구의 중요성”, (주)시큐아이닷컴 CERT팀, 2007.

[6] 김도완, 윤영선, “SW소스코드 저작권보호를 위한 통합 가이드”, 컴퓨터프로그램보호위원회, April, 2009.

[7] 길연희, 홍도원, “디지털 포렌식 기술과 표준화 동향”, IT standard & test TTA journal, Aug. 2008,

[8] 변정수, “한국형 디지털 증거분석 표준화: 경찰청 디지털 증거처리 표준가이드라인 및 증거분석 전문매뉴얼의 고찰”, 디지털 포렌식 연구 창간호, Nov. 2007.

[9] 방효근, 신동명, 정태명, “소프트웨어 포렌식: 프로그램 소스코드 유사성 비교 및 분석을 중심으로”, 디지털 포렌식 연구 창간호, Nov. 2007.

[10] 이규대, “임베디드시스템의 이진코드 추출 및 분석”, 한국소프트웨어감정평가학회 논문지, 5권1호, pp.27-38, May, 2009.

[11] 전병태, “프로그램 복제도 감정기법 및 감정비 산출에 관한 연구”, 프로그램심의조정위원회 결과보고서, 2002.

[12] 이규대, 권기영, “정보기기 감정에서 세부항목 설정 사례”, 한국소프트웨어감정평가학회 논문지, 12권2호, pp.9-14, Dec. 2016.

저 자 소 개

이규대(Kyu-Tae Lee)

1991 고려대 전자공학과 박사
2001 미 조지아텍 교환 교수
2006 미 일리노이주립대 교환 교수
‘2007～2009: ETRI 이동통신소 초빙연구원
‘1992.3～현재: 공주대 정보통신공학부 교수
<주관심분야> 회로 및 시스템, 신호처리, VLC, 저작권보호