FUSE 기반의 디바이스 접근 로그 기법

이희재*,임성락(교신저자)*
*호서대학교 컴퓨터공학과
e-mail:e08y33@naver.com

A Scheme of Device Access Log based on FUSE

Hui-Jae Lee*, Seongrak Rim*
*Computer Engineering, Ho-Seo University

요 약

IOT의 발전으로 많은 디바이스들이 사용자들에게 제공되어 데이터를 얻거나 제어함으로서 유용하게 사용되고 있다. 보통 하나의 디바이스에 다수의 사용자가 접근하여 데이터를 얻거나 제어 함으로써 서비스를 받게 된다. 그러나 하나의 디바이스를 여러 사용자가 이용하게 되면 어떤 사용자에 의해 디바이스가 오작동을 할 수 있고 결함이 발생할 수가 있다. 본 논문에서는 이러한 경우를 예방하고 디바이스를 관리하기 위해 FUSE와 Syslog를 이용하여 디바이스 접근에 대한 정보 로그하는 기법을 제시한다. 이를 위하여 리눅스(Ubuntu 16.04)에서 문자 디바이스 드라이버 모듈을 작성하여 커널에 삽입하고, 디바이스에 접근을 시도하는 테스트 프로그램을 작성하여 디바이스에 접근할 때 접근 정보를 로그하는 기법을 제시한다.

1. 서론

IOT의 발전으로 많은 디바이스들이 사용자들에게 제공되어 사용되고 있다. 그러나 다수의 사용자가 하나의 디바이스를 사용하게 되면 디바이스의 오작동, 결함이 발생할 가능성이 높아진다. 이러한 경우를 관리자로 하여금 예방하고 해결하기 위하여 디바이스 접근에 대한 로그를 남겨 문제를 해결한다. 본 논문에서는 FUSE(Filesystem in Userspace)를 이용한 문자 디바이스 파일을 생성하고 Syslog를 이용해 디바이스에 접근하는 사용자에 대한 정보를 로그 하는 기법을 제시한다.

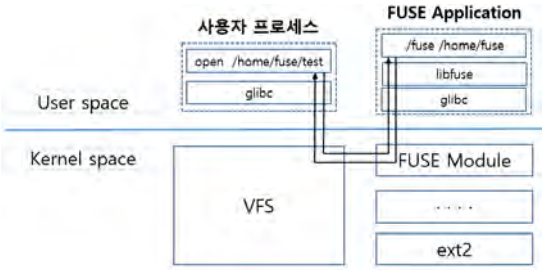
이를 위하여 FUSE에서 관리하는 디바이스 파일을 통해 사용자가 디바이스에 접근할 때 사용자에 대한 ID, 접근 시간을 Syslog에 전달하고 로그파일에 기록한다.

2장에서 FUSE의 동작 원리와 리눅스에서 문자 디바이스 접근 과정, syslog에 대한 관련 연구를 기술한다. 3장에서는 본 논문에서 제시한 기법의 타당성 검토를 하기 위한 구현 및 실험을 기술한다. 4장에서는 제시한 기법의 결과와 향후 연구 과제를 제시한다.

2. 관련연구

2.1 FUSE(Filesystem in Userspace)

FUSE는 (그림 1)과 같이 커널에 있는 FUSE 모듈(FUSE Module)과 사용자 수준의 라이브러리(libfuse)로 구성되어 있다[1,2].



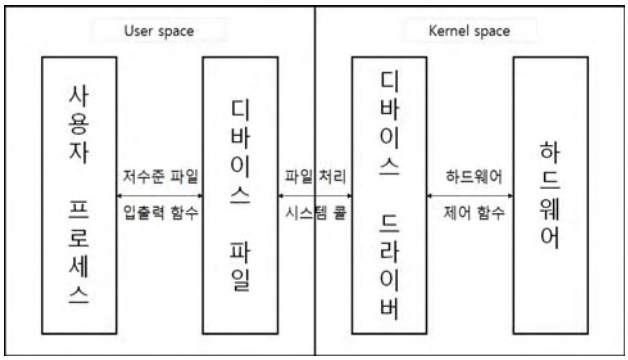
(그림 1) FUSE 동작 흐름

(그림 1)에서 사용자 프로세스는 open()을 호출하여 open() 시스템 콜을 발생시킨다. open() 시스템 콜은 VFS(Virtual File System)이 받아 어떤 파일시스템의 파일을 열기 위한 요청인지 확인한다. FUSE의 파일 열기를 요청하면 FUSE 모듈을 통해 FUSE Application으로 전달한다. FUSE Application은 전달받은 열기 요청을 처리하고 FUSE 모듈을 통해 VFS에 전달함으로써 사용자 프로세스는 open() 호출에 대한 결과를 받게 된다.

이와 같이 FUSE를 이용하여 사용자 영역에서 동작되는 파일 시스템을 구현할 수 있으며 FUSE를 통하여 생성된 파일들에 대한 접근은 FUSE Application에서 구현된 파일 오퍼레이션에 의해 처리된다. 따라서 파일 시스템의 기능을 사용자 영역에서 구현할 수 있기 때문에 커널 내부의 모듈 프로그래밍보다 용이하다. 또한 사용자 수준의 다양한 라이브러리 함수를 이용할 수 있어 새로운 기능 추가 및 변경이 보다 편리한 장점이 있다[3,4,5]

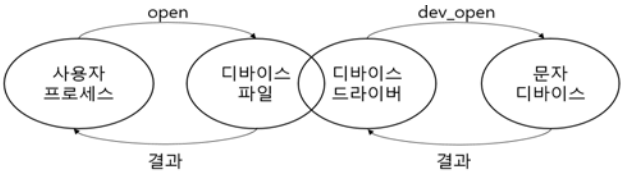
2.2 문자 디바이스 접근

디바이스를 접근하기 위해 사용자는 (그림 2)와 같이 저수준 파일 입출력 함수를 이용해 디바이스 파일에 데이터를 읽고 쓴다[6]. 디바이스 드라이버는 디바이스 파일과 연결되어 응용프로그램과 하드웨어 간의 데이터를 주고받을 수 있는 역할을 한다.



(그림 2) 디바이스 접근 과정

일반적으로 리눅스에서 문자 디바이스를 접근하는 과정은(그림 3)와 같이 디바이스 파일을 통하여 디바이스에 접근한다.



(그림 3) 문자 디바이스 접근 과정

(그림 3)에서 문자 디바이스에 접근하기 위해서는 사용자 프로세스에서 디바이스 파일에 대한 open() 시스템 콜을 호출한다. 디바이스 파일에 대한 open() 요청은 커널 안의 디바이스 드라이버로 전달된다. 디바이스 드라이버는 전달받은 open() 함수에 맵핑된 dev_open() 함수를 호출하여 문자 디바이스에 접근할 수 있는 파일 디스크립트 번호를 응용 프로그램에 반환한다[7].

2.3 Syslog

수많은 시스템의 커널 경고, 디버깅 정보, 각종 메시지 출력 등의 다양한 활동에 대한 정보를 받아서 일괄로 처리, 기록하는 데몬 프로그램이다. Syslog는 <표 1>,<표 2>에 보이는 메시지 기능 코드와 메세지 등급 코드를 이용해 로그 하기 위한 정보의 종류와 등급을 설정하여 로그 메시지의 종류와 등급을 설정하고 이에 해당하는 정보를 로그 파일에 기록한다[8]. 관리자는 Syslog를 통해 기록된 로그파일을 확인함으로써 시스템의 이상 유무를 점검한다.

<표 1> Syslog 메세지 기능 코드

번호	설명
0	커널 메세지
1	사용자 계층 메세지
2	메일 시스템
3	시스템 데몬
4	보안/허가 메세지
5	syslogd에 의해 내부적으로 생성된 메세지
6	라인 프린터 서브 시스템
7	네트워크 뉴스 서브 시스템
8	UUCP서브 시스템
9	클락 덱몬
10	보안/허가 메세지
11	FTP 데몬
12	NTP 서브 시스템
13	로그 감사
14	로그 경보
15	클락 데몬
16~23	지역 사용 0 ~ 지역 사용7

<표 2> Syslog 메세지 등급 코드

코드	등급	설명
0	Emergency	시스템 사용불가
1	Alert	즉시 조치
2	Critical	치명적인 상태
3	Error	오류 상태
4	Warning	주의 상태
5	Notice	정상이지만 중요한상태
6	Informational	정보 메시지
7	Debug	디버그 수준의 메세지

3. 구현 및 실험

3.1 구현

제시한 기법의 기능적 타당성을 검토하기 위하여 FUSE Application의 main() 함수와 파일 오퍼레이션(test_oper)을 (그림 4)과 같이 구현한다.

(그림 4)에서 main() 함수는 FUSE 파일시스템이 Root 파일 시스템에 마운트 시킬 때 필요한 디렉터리명 및 옵션 등을 인수로 전달받아 fuse_main()을 호출한다.

openlog() 함수는 syslog() 함수를 호출하여 로그를 기록하기 전에 system logger와 연결하기 위해 호출한다. openlog()에 사용되는 인자는 첫 번째 인자(“FUSE Application”) FUSE Application에서 생성된 인자를 확인하기 위해 사용한다. 두 번째 인자(LOG_PID)는 각각의 로그 메시지에 PID를 포함시킨다. 세 번째 인자(LOG_LOCAL1)는 로그 메시지를 로컬 1번으로 출력하기 위한 목적으로 사용된다.

```
static struct fuse_operations syslog_oper = {
        //파일의 속성을 얻는 오퍼레이션
        .getattr = syslog_getattr,
        // 디렉터리를 읽는 오퍼레이션
        .readdir = syslog_readdir,
        // 디바이스 파일 생성 오퍼레이션
        .mknod = syslog_mknod,
        // 파일 open 오퍼레이션
        .open = syslog_open,
};
int main(int argc, char *argv [])
{
        openlog("FUSE Application",LOG_PID,
                                        LOG_LOCAL1);
        //FUSE Application과 system logger를 연결
          return fuse_main(argc, argv,
                                &syslog_oper, NULL);
}
```

(그림 4) FUSE_operations 및 main()

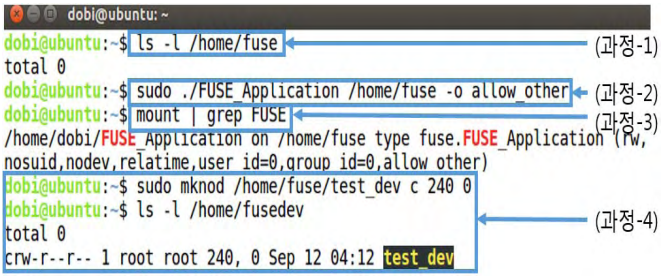
(그림 5)사용자가 디바이스 파일에 접근할 때 "fuse_get_context()->uid"의 값을 통해 getpwuid()를 호출하여 사용자의 정보를 구해 passwd 구조체에 저장한다. getpwuid()를 통해서 얻어온 사용자의 정보를 syslog() 함수를 이용하여 디바이스 접근 정보를 기록한다.

```
static int syslog_open()
{
        int res = 0;
        char device_file[1024];
        struct passwd *cur_user =
                        getpwuid(fuse_get_context()->uid);
        // 디바이스에 접근하는 사용자 정보를 가져온다.
        res = open(device_file,fi->flags);
        syslog(LOG_INFO | LOG_LOCAL1, "%s ->
                device file open()", cur_user->pw_name);
        // syslog()를 이용해 접근 정보를 기록한다.
        if(res<0) return -errno;
        close(res);
        return 0;
}
```

(그림 5) FUSE 파일 시스템의 open()

3.2 실험

제시한 기법의 타당성을 검토하기 위하여 다음과 같은 과정으로 실험한다.



(그림 6) FUSE 파일시스템 마운트

[과정1] FUSE 파일 시스템을 마운트 시킬 디렉터리를 확인한다.

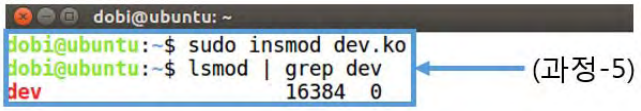
[과정2] 마운트 디렉터리를 지정하고 FUSE_Application을 실행한다.

[과정3] FUSE 파일 시스템이 마운트되어 있는지 확인한다.

[과정4] 디바이스 파일을 생성하고 디바이스 파일이 생성됨을 확인한다.

```
#define MAJOR_NUMBER 240
static int test_open() {
  printk("[test_open]\n"); // 커널 메시지 출력
  return 0;
}
static ssize_t test_release() {
  printk("[test_release]\n"); //커널 메시지 출력
  return count;
}
static struct file_operations dev_fops = {
  .owner=THIS_MODULE,
  .open=test_open,
  .release=test_release,
};
int test_init(){ · · · }
int test_exit(){ · · · }
```

(그림 7) 디바이스 드라이버

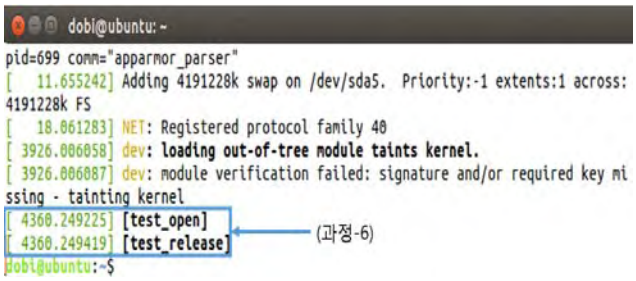


(그림 8) 디바이스 드라이버 모듈 삽입

[과정5] 디바이스 접근 확인을 위한 디바이스 모듈 삽입한다.

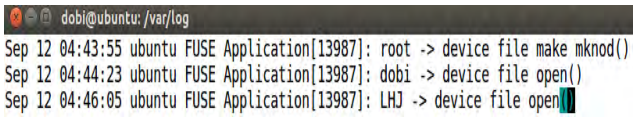
```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
int main(void) {
  int fd ;
  fd = open("/home/fuse/test_dev",
                    O_RDWR | O_NDELAY);
  return 0;
}
```

(그림 9) 테스트 프로그램



(그림 10) 커널 메시지 확인

[과정6] (그림 9)의 테스트 프로그램을 통해 디바이스에 접근하고 정상적으로 동작하는지 확인한다(그림 10).



(그림 11) 로그 파일

[과정7] 디바이스 접근에 대한 log파일(그림 11)을 확인한다.

4. 결론

본 논문에서는 다수의 사용자가 하나의 디바이스를 사용할 때 사용자에 대한 정보를 로그로 남기는 기법을 제시했다. 디바이스에 문제가 생겼을 때 디바이스 관리자로 하여금 디바이스에 대한 문제를 빠르게 확인하고 조치하게 함으로써 디바이스의 유지 보수를 수월하게 한다. 제시한 기법을 확인하기 위해 사용자 영역에서 구현된 FUSE 파일시스템을 이용해 디바이스 파일을 생성하고 사용자로 하여금 FUSE 파일시스템이 관리하는 디바이스 파일을 통해 디바이스에 접근할 때 사용자 정보를 Syslog를 이용하여 로그로 남기는 기법을 제시하였다. 제시한 디바이스 접근 로그 기법을 확인하기 위해 리눅스 (Ubuntu 16.04) 운영체제를 사용하여 디바이스 파일을 생성하고 이를 통해 디바이스 드라이버에 정상적으로 접근, 디바이스 접근 정보를 로그 하는 것을 확인함으로써 기능적 타당성을 검토하였다.

FUSE 파일 시스템과 Syslog를 사용하여 디바이스에 대한 접근 정보를 로그할 수 있음을 확인하였다. 향후 FUSE 파일시스템과 syslog를 사용하여 디바이스에 접근하는 사용자의 정보를 더 다양하고 자세하게 수집하여 로그 하는 방법에 대한 연구가 필요하다.

참고문헌

[1] https://github.com/libfuse/libfuse

[2] 김문경, 엄현철, 노재춘, 박성순, “FUSE(Filesystem in Userspace) 기반 WROM(Write Once Read Many) 파일 시스템의 설계 및 구현”, 한국정보과학회, 2008가을 학술발표논문집, 제35권 제2호(B), 2008.10, p395-400

[3] 김수영, 김홍연, 김영균, “FUSE 활용 pNFS 지원 메타데이터 서버 설계”, 2012 한국컴퓨터종합학술대회 논문집, 제 39권 제 1호(A), 2012.6, p1-3

[4] 이희재, “FUSE를 이용한 문자 디바이스 접근 기법”, 호서대학교 공업기술연구 논문지

[5] 손태영, “파일 접근 로그를 위한 FUSE 기반의 syslog 에이전트”, 한국산학기술학회 논문지, 제 17권 제7호, 2016

[6] 노성동, 이명의, “임베디드 리눅스 구조 및 응용, GS인터비전(2009), p217-219

[7] 유영창, 리눅스 디바이스 드라이버, 한빛미디어(2004), p164-165

[8] 황훈규, 윤진식, 서정민, 이성대, 장길웅, 박휴찬, 이장세, “이더넷 기반 선박 통합 네트워크를 위한 로그 처리 모듈 및 로그 서버의 개발