Kernel Module Programming

Kernel Module

- Linux Kernel
  - ✓ Monolithic Kernel (단일형 커널)
    - ➢ 운영 체제의 기능의 거의 모든 것이 단일 메모리 공간에서 수행
  - ✓ 커널 기능 확장 (e.g., 디바이스 추가)
    - ➢ 커널 컴파일이 요구됨
- Kernel Module
  - ✓ 커널을 확장하기 위해 사용하는 오브젝트 모듈
  - ✓ 커널이 실행 중에 동적으로 로딩하여 커널과 링크함으로써 커널의 기능을 확장하여 사용할 수 있으며, 불필요 시에 커널과의 링크를 풀고 메모리에서 제거할 수 있음
    - ➢ 커널 컴퍼일 없이 기능 확장 가능

Kernel Module

- Kernel Module
  - ✓명시적인 커널 모듈 설치 및 제거 과정이 필요
    - ➢ insmod / rmmod 명령어
  - ✓디바이스 드라이버, 파일시스템, 네트워크 프로토콜 스택 등에 적용
    - ➢커널 경량화를 위해 반드시 필요
    - ➢임베디드 시스템: 제한적인 자원으로 인해 커널 등 시스템 소프트웨어의 최소화가 필요

Kernel Module Programming

- 커널 모듈은 커널 모드에서 실행됨으로 커널 프로그래밍에 준하여 작성
- 주의 사항
  - ✓ 무제한적인 메모리 접근 ➔ 메모리 접근 오류는 시스템에 치명적인 손상을 줄 수 있음
  - ✓ 커널 함수 호출에 따른 에러 코드 검사
  - ✓ 실수 연산 및 MMX 연산 사용 불가
  - ✓ 제한적인 커널 스택 크기
    - ➢ 크기가 큰 배열 사용 회피 ➔ 동적 할당 방법을 이용
    - ➢ 재귀 호출 회피
  - ✓ 플랫폼 고려
    - ➢ 32 bit 및 64 bit 환경
    - ➢ 바이트 순서 – big-endian 또는 little-endian
    - ➢ 프로세서에 특화된 명령어 사용 회피
  - ✓ 버전 종속성 - 커널 버전에 따라 모듈 버전이 다름

Kernel Version Dependency

- 커널 버전에 따라 지원되는 커널 함수의 종류나 프로토타입이 다르기 때문에 다른 버전의 커널에 모듈을 링크하려면 모듈 재컴파일이 필요

- 모듈을 커널에 설치할 때에 버전 검사 수행
  - ✓ 커널 버전과 모듈 버전이 일치하지 않는 경우에 모듈을 설치할 수 없음

- 커널 모듈 프로그램의 버전 정의
  - ✓ 커널 ver.2.0 이상에서는 <linux/module.h> 헤드 파일에서 char kernel_version[ ] 변수에 커널 버전을 정의
  - ✓ 커널 모듈 프로그램에서 위의 헤드 파일을 including
    #include <linux/module.h>
  - ✓ 커널 버전 정의는 전체 모듈에서 한번만 정의되어야 함

Kernel Module Program

- 커널 모듈 구성
  - ✓ 커널 기능을 확장한 함수와 자료 구조로 구성
    - ➢ main() 함수는 없는 독립된 프로그램 모듈
  - ✓ init_module()/cleanup_module() 함수 사용
    - ➢ init_module() 함수: 모듈이 설치될 때에 자동적으로 호출 / 모듈 초기화 등의 기능 수행
    - ➢ cleanup_module() 함수: 모듈을 삭제할 때에 자동적으로 호출 / 모듈 제거에 따른 cleanup 작업을 수행

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
/* global variables */
…
int init_module(void) { }
void cleanup_module(void) { }
…
```

Kernel Module Program

- 커널 모듈 구성
  - ✓ 커널 ver.2.4 이상에서는 module_init() / module_exit() 매크로 지원
    - ➢ 함수 이름에 의한 종속 관계를 해결
    - ➢ module_init() 매크로 : startup 함수 등록
    - ➢ module_exit() 매크로 : cleanup 함수 등록

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
/* global variables */
…
int module_start() { /* 모듈이 설치될 때에 초기화를 수행하는 코드 */ }
int module_end() { /* 모듈이 제거될 때에 반환작업을 수행하는 코드 */ }
module_init(module_start);
module_exit(module_end);
…
```

Kernel Module Program

- 모듈 프로그램의 Makefile
  - ✓ -c : 링커를 호출하지 않고 오브젝트 모듈만 생성
  - ✓ -O : 최적화 옵션 지정 (최대 O2)
  - ✓ 컴파일 매크로 : __KERNEL__ , MODULE, LINUX

```
CC = arm_linux_gcc
INCLUDEDIR = /kernel_source_dir/include
CFLAGS := -c -O -Wall -D__KERNEL__ -DMODULE -DLINUX -I$(INCLUDEDIR)
OBJS = hello.o
all : $(OBJS)
clean : rm -f *.o *
```

커널 프로그램 임을 명시하는 매크로

모듈 프로그램 임을 명시하는 매크로

Linux 플랫폼에서 동작하는 모듈임을 정의하는 매크로

Kernel Module Program

- 커널 모듈 명령어

이 름	용 도
insmod	**module**을 설치**(install)**
rmmod	실행중인 **modules**을 제거**(unload)**
lsmod	**Load**된 **module**들의 정보를 표시
depmod	**Module**들의 **symbol**들을 이용하여 **Makefile**과 유사한 **dependency file**을 생성
modprobe	**depmod**명령으로 생성된 **dependency**를 이용하여 지정된 디렉토리의 **module**들과 연관된 **module**들을 자동으로 **load**
modinfo	목적 파일을 검사해서 관련된 정보를 표시

Kernel Module Program

- 커널 모듈 설치 및 제거
  - ✓ 설치된 모듈은 '/proc/modules' 파일에 기록
  - ✓ '/proc/modules' 파일을 이용한 현재 설치된 모듈을 확인 가능
    #cat /proc/modules

Kernel Module Program (실습)

- 커널 모듈 프로그램 작성
  - ✓ 모듈 적재 및 제거 시에 간단한 메시지 로그를 출력하는 모듈 작성
  - ✓ 구현 방법
    - ➢ 모듈 적재 시에 호출되는 init_module() 함수에서 printk() 함수를 이용하여 모듈 로딩 (loading) 메시지를 출력
    - ➢ 모듈 제거 시에 호출되는 cleanup_module() 함수에서 같은 방법으로 모듈 언로딩 (unloading) 메시지를 출력
  - ✓ 모듈 작성 및 테스트
    - ➢ 모듈 프로그램 소스 파일 hello_m.c을 작성

Kernel Module Program (실습)

- 커널 모듈 소스(hello_m.c)

```
#include <linux/module.h> /* Needed by all modules */
#include <linux/kernel.h> /* Needed for KERN_ALERT */

int init_module(void){
    printk("<1>hello module loaded\n");

    // A non 0 return means init_module failed
    return 0;
}

void cleanup_module(void){
    printk(KERN_ALERT "hello module unloaded\n");
}
```

Kernel Module Program (실습)

- 함수 printk()의 인수 :
  - ✓ File "/lib/modules/`uname –r`/build/include/linux/kernel.h"에 정의

```
#define KERN_EMERG    "<0>"  /* system is unusable */
#define KERN_ALERT    "<1>"  /* action must be taken immediately  */
#define KERN_CRIT     "<2>"  /* critical conditions */
#define KERN_ERR      "<3>"  /* error conditions */
#define KERN_WARNING  "<4>"  /* warning conditions */
#define KERN_NOTICE   "<5>"  /* normal but significant condition */
#define KERN_INFO     "<6>"  /* informational */
#define KERN_DEBUG    "<7>"  /* debug-level messages */
```

Kernel Module Program (실습)

- module_init/module_exit 사용

```
#include <linux/module.h>    // Needed by all modules
#include <linux/kernel.h>    // Needed for KERN_ALERT
#include <linux/init.h>          // Needed for the macros

static int hello_init(void){
  printk(KERN_ALERT “hello module loaded\n”);
  return 0;
}

static void hello_exit(void){
  printk(KERN_ALERT “hello module unloaded\n”);
}

module_init(hello_init);
module_exit(hello_exit);
```

Kernel Module Program (실습)

- Module Compile

```
TARGET = hello_m
WARN = -Wall -Wstrict-prototypes -Wmissing-prototypes
INCLUDE = -isystem /lib/modules/`uname -r`/build/include
CFLAGS = -O2 -DMODULE -D__KERNEL__ ${WARN} ${INCLUDE}
CC = arm_linux_gcc

$(TARGET).o: $(TARGET).c

clean:
    rm -rf $(TARGET).o
```

Kernel Module Program (실습)

- 모듈 'hello_m.o'을 다운로드하고 다음과 같이 테스트한다
  - ✓ 모듈 load 명령 줄

```
# insmod hello.o
```

  - ✓ 모듈 load 확인 명령 줄

```
# lsmod
Module                  Size Used by
hello                   368  0  (unused)
```

  - ✓ 모듈 unload 명령 줄

```
# rmmod hello
```

  - ✓ printk() 메시지 확인

```
# tail -2 /var/log/messages
Oct  12  17:54:29 kernel: hello module loaded
Oct  12  17:54:45 kernel: hello module unloaded
```

E N D