Operating System 실습
[Linux Module]

한양대학교
HANYANG UNIVERSITY

Data-Centric Systems Lab.

Background

- Monolithic kernel
  - 커널 설정을 변경한 경우 커널 전체가 다시 컴파일 되어야 함.
  - 자주 사용되지 않는 파일 시스템이나 드라이버 코드가 메모리에 상주하게 된다.
  - 커널 코드를 변경하면 컴파일을 다시 해야 할 뿐만 아니라 매번 재부팅이 필요하다.

- Linux Module
  - 파일 시스템이나 드라이버 같은 커널의 funnctional unit
  - 시스템이 부팅된 후라 할지라도 언제든지 커널과 동적으로 링크됨.
  - 동적으로 적재된 모듈도 커널의 일부로 취급됨.
  - 더 이상 필요하지 않으면 언제든지 제거하여 커널 메모리 낭비를 막을 수 있음.

Macros

- MODULE_LICENSE()

GPL	GNU Public License v2 or later
GPL v2	GNU Public License v2
GPL and additional rights	GNU Public License v2 rights and more
Dual BSD/GPL	GNU Public License v2 or BSD license choice
Dual MIT/GPL	GNU Public License v2 or MIT license choice
Dual MPL/GPL	GNU Public License v2 or Mozilla license choice

- MODULE_DESCRIPTION()
  - is used to describe what the module does
- MODULE_AUTHOR()
  - declares the module's author
- MODULE_SUPORTED_DEVICE()
  - declares what types of devices the module supports

Modules Command

- Modules Shell Command

명령어	설명
insmod	모듈 install
rmmod	실행중인 모듈 제거
lsmod	Load 된 모듈의 정보 표시
modinfo	Ko 파일의 관련 정보 표시
depmod	커널 내부에 적재된 모듈간의 의존성 검사
modeprobe	의존성을 검사하여 누락된 모듈을 적재

실습 (hello.c)

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

static int hello_init(void)
{
    printk(KERN_ALERT "I bear a charmed life.\n");
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_ALERT "Out, out, brief candle!\n");
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A hello, World Module");
MODULE_AUTHOR("Kihan Choi");
```

실습 (Makefile)

```
obj-m   := hello.o

KDIR    := /lib/modules/$(shell uname -r)/build
PWD     := $(shell pwd)

default:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf .*.cmd
	rm -rf *.o
```

실습

- $dmesg –c //clear dmesg
- $ make
- $modinfo hello.ko //module 정보 출력
- $ insmod hello.ko
- $ lsmod
- $ rmmod hello.ko
- $ dmesg

Function

- hello_init()
  - Registered as this module's entry point via module_init()
  - The kernel invokes when the module is loaded

- module_init()
  - Not actually a function call
  - A macro that assigns its sole parameter as the initialization function
  - All init functions must have the form
    - Int my_init(void)

- module_exit()
  - Registers a module's exit point
  - The kernel invokes when the module is removed from memory
  - Responsible for undoing whatever the init function and lifetime of the module did
  - Exit functions must have the form
    - Int my_exit(void)

Debugging by printing

- printk()
  - Happens to be a logging mechanism for the kernel, and is used to log information or give warnings
  - printk message is stored to ring buffer

- Loglevels
  - Declared in <linux/kernel.h>

Loglevel	Description
KERN_EMERG	An emergency condition; the system is probably dead.
KERN_ALERT	A problem that requires immediate attention.
KERN_CRIT	A critical condition.
KERN_ERR	An error.
KERN_WARNING	A warning.
KERN_NOTICE	A normal, but perhaps noteworthy, condition.
KERN_INFO	An informational message.
KERN_DEBUG	A debug message—typically superfluous.

실습 (param_mod.c)

```c
#include<linux/init.h>
#include<linux/module.h>
#include<linux/kernel.h>


static int age = 0;
static char *name = NULL;


module_param(name,charp,0);
module_param(age,int,0);

static int hello_init(void)
{
    printk(KERN_ALERT "hello, My name is %s, my age is %d\n", name, age);
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_ALERT "Bye bye!\n");
}


module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Module Parameters");
MODULE_AUTHOR("Kihan Choi");
```

Module Parameters

- Creating and managing module parameters that can be specified in a myriad of convenient ways is trivial

- module_param();
  - Define a module parameter

```
module_param(name, type, perm);
```

    * Name

      The name of both the parameter exposed to the user and the variable holding the parameter inside your module

    * Type

      Holds the parameter's data type

      Byte, short, ushort, int, uint, long, ulong, charp, bool, invbool

    * Perm

      The permissions of the corresponding file in sysfs

실습

- Modify Makefile
- $dmesg -c
- $ make
- $modinfo param_mod.ko
- $ insmod param_mod.ko name="Kihan" age=15
- $ rmmod param_mod.ko
- $ dmesg

실습 결과

```
os@os-VirtualBox:~/practice$ modinfo hello.ko
filename:       hello.ko
author:         Kihan Choi
description:    A hello, World Module
license:        GPL
srcversion:     D78D32956BF669CDE6E010D
depends:
vermagic:       3.11.0 SMP mod_unload modversions 686
```

```
os@os-VirtualBox:~/practice$ modinfo param_mod.ko
filename:       param_mod.ko
author:         Kihan Choi
description:    Module Parameters
license:        GPL
srcversion:     7EE06BDF4C3486659C8C461
depends:
vermagic:       3.11.0 SMP mod_unload modversions 686
parm:           name:charp
parm:           age:int
```

```
os@os-VirtualBox:~/practice$ dmesg
[147442.116355] hello, My name is Kihan, my age is 15
[148433.209435] Bye bye!
```