



Operating System Practice

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University



Lab2: Build a Kernel Module on OMAP 5912

Rebuild the Environment (1 / 2)

- ▶ Prepare the root filesystem
 - *su* (the password is 123456)
 - *export PATH=\$PATH:/opt/usr/local/arm/3.3.2/bin*
 - Copy the downloaded rootfsosk.tar.bz2 to
/tmp/rootfsosk.tar.bz2
 - *cd /tmp*
 - *tar jxvf rootfsosk.tar.bz2*
- ▶ Disable the Firewall (it is not a good idea, only for lab)
 - *systemctl stop firewalld*
 - *systemctl disable firewalld*

Rebuild the Environment (2/2)

▶ Set the TFTP and NFS Service

- ~~/sbin/chkconfig xinetd on~~
 - *systemctl start tftp.socket*
 - ~~/sbin/service xinetd start~~
 - *systemctl enable tftp.socket*
 - *exportfs -rv*
 - *systemctl start rpcbind.service*
 - *systemctl start nfs-mountd.service*

A Simple Linux Module

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("License for you");

static int mymodule_init(void)
{
    printk("Insert My Module to the Linux Kernel!\n");
    return 0;
}

static void mymodule_exit(void)
{
    printk("My Module is Unloaded!\n");
}

module_init(mymodule_init);
module_exit(mymodule_exit);
```

The diagram consists of five callout boxes with red borders and arrows pointing to specific parts of the code. The first box, 'The License', points to the `MODULE_LICENSE` line. The second box, 'Initial Function', points to the `mymodule_init` function. The third box, 'Exit Function', points to the `mymodule_exit` function. The fourth box, 'Let the Kernel Know the Functions', points to the two `module_*` lines at the bottom of the code.

A Simple Makefile

```
ifneq ($(KERNELRELEASE),)
    obj-m :=testmod.o
else
    KDIR :=/opt/linux-2.6.12
    PWD :=$(shell pwd)
    CC :=/opt/usr/local/arm/3.3.2/bin/arm-linux-gcc
default:
    $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
endif
```

Download and Compile Files

- ▶ Download the Makefile and the kernel module source code
- ▶ Read and understand the files
- ▶ Compile the kernel module: *make*
- ▶ Copy the kernel module (testmod.ko) to the NFS file system: *cp testmod.ko /tmp/rootfs2.6*
- ▶ Connect to the evaluation board by minicom

Rebuild Connection

- ▶ Start another terminal
- ▶ Minicom
 - *su*
 - *minicom*
 - If your minicom is in Chinese: *export LANG=en*
- ▶ Set the boot configuration
 - *set ipaddr 192.168.68.yy* (evaluation board IP)
 - *set serverip 192.168.68.kk* (PC IP)
 - *set netmask 255.255.255.0*
 - *set gatewayip 192.168.68.254*
 - *set ethaddr 00-0e-99-xx-xx-xx* (the MAC address)
 - *set bootargs console=ttyS0,115200n8 rw ip=192.168.68.yy root=/dev/nfs nfsroot=192.168.68.kk:/tmp/rootfs2.6,v3*
 - *printenv* → double check the setting
 - *saveenv* → if everything is correct → be careful, do not crash the entire system
 - Download the kernel: *tftpboot 0x10000000 ulimage* (capital i)
 - Boot the OS: *bootm 0x10000000*

Check Point 1

- ▶ Use the following command to insert the kernel module
 - *insmod testmod.ko*
- ▶ Use the following command to unload the kernel module
 - *rmmmod testmod*
- ▶ Check the result on the screen
 - For your Linux desktops, you might need to use the command *dmesg*, but you don't need to do it in this exercise

Control Devices by Kernel Modules

- ▶ Control the LEDs on the evaluation board
 - Include the header file: `#include <asm-arm/arch-omap/tps65010.h>`
 - Use the function: `tps65010_set_led(which_LED , what_operation);`
- ▶ The LEDs on the board
 - LED1
 - LED2
- ▶ The functions of a LED
 - OFF
 - ON
 - BLINK
- ▶ An example: make LED 1 blink
 - `tps65010_set_led(LED1 , BLINK);`

Check Point 2

- ▶ Modify the kernel module
 - Make LED 1 “ON” when the module is loaded
 - Make LED 1 “OFF” when the module is unload
- ▶ Can the LED be directly modified by user applications?

Timer

- ▶ **jiffies** is the counter for the timer interrupts
- ▶ **HZ** is the number of timer interrupts per second
- ▶ An example to BLINK for 3 seconds and then be ON
 - `#include <linux/timer.h>` // you have to include the header file
 - `int a;` // it should be declared outside as a global variable here
 - Using it
 - `tps65010_set_led(LED1, BLINK);`
 - `a = jiffies;`
 - `while(a + 3*HZ > jiffies) ;`
 - `tps65010_set_led(LED1, ON);`

Check Point 3

- ▶ Modify the kernel module
 - The procedure of the module insertion
 - Make LED 1 “BLINK” at the beginning of the insertion
 - The duration of BLINK is 7 seconds
 - After the insertion, make LED 1 “ON”
 - The procedure of the deletion
 - Make LED 1 “BLINK” at the beginning of the deletion
 - The duration of BLINK is 5 seconds
 - After the deletion, make LED 1 “OFF”

Grading this Exercise

- ▶ Attend and understand this exercise: 10%
- ▶ Check point 1: 10%
- ▶ Check point 2: 10%
- ▶ Final results: 20%
- ▶ Report before the exercise: 25%
- ▶ Report after the exercise: 25%

Report Requirements

- ▶ Report before the exercise: 25%
 - Only two pages, 12-pt font size
 - Deadline is 20:00, 2019/06/05
 - File name: OSP-Lab2-Study-StudentID
 - File type: PDF or Word
 - Send it to my email: kf200660306@gmail.com
 - Email title: OSP Lab2 Study StudentID
- ▶ Report after the exercise: 25%
 - Only two pages, 12-pt font size
 - Deadline is 20:00, 2019/06/12
 - File name: OSP-Lab2-Report-GroupID
 - File type: PDF or Word
 - Send it to my email: kf200660306@gmail.com
 - Email title: OSP Lab2 Report GroupID
 - Remember to list all student IDs of your group