Embedded Operating System

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("Instert 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 License

Initial Function

Exit Function

Let the Kernel Know the Functions

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 uImage* (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
  - *rmmod 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

Common Mistakes

- *su* and *export* should be used whenever a new terminal is created
  - If you extract the root file system by the user csie, there will be an error when you boot the board to mount the NFS root file system
    - Reboot the computer and do everything again
  - If you do not export the path of the tools, you will get some error when you compile the kernel module
- Please read the error message if you type something wrong
- UART: it should be connected to the bottom port
- Ethernet: do check the IP is correct

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
  - 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 the insertion state is 10 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 the insertion is 7 seconds
    - After the insertion, make LED 1 “OFF”

Grading this Exercise

- Attend and understand this exercise: 20%
- Check point 1: 10%
- Check point 2: 10%
- Check point 3: 10%
- Report before the exercise: 20%
  - Only one page A4, 12 pt font
  - Deadline is 20:59 2019/12/30
  - File name: EOS-Lab2-Study-Student_ID
  - File type: PDF or Word
  - Send it to my email: chewei@mail.cgu.edu.tw
  - Email title: EOS Lab2 Study Student_ID
- Report after the exercise: 30%
  - Only one page A4, 12 pt font
  - Deadline is 20:59 2020/1/6
  - File name: EOS-Lab2-Report-Student_ID
  - File type: PDF or Word
  - Send it to my email: chewei@mail.cgu.edu.tw
  - Email title: EOS Lab2 Report Student_ID
- Bonus: 30%
  - Study the rules of Makefile and summarize them in your report before the exercise