浙江工业大学

实验报告

课程：嵌入式系统 A

第五次实验

姓　　名	**凌智城**
学　　号	**201806061211**
专业班级	**通信工程 1803 班**
老　　师	**黄国兴**
学　　院	**信息工程学院**

提交日期　2021 年 6 月 3 日

实验 8：RTC 时钟驱动实验

一、 实验目的

1. 了解 RTC 工作原理；
2. 掌握 RTC 驱动的编写；
3. 掌握 RTC 的加载过程及测试方法。

二、 实验内容

1. 学习 RTC 的工作原理；
2. 编写 RTC 的驱动程序；
3. 编写测试程序测试 RTC。

三、 实验步骤

步骤 1：建文件夹，编写 RTC 驱动程序（在服务器或虚拟机终端运行）

创建 rtc 文件夹，并将服务器中的 RTC 文件夹复制过来。

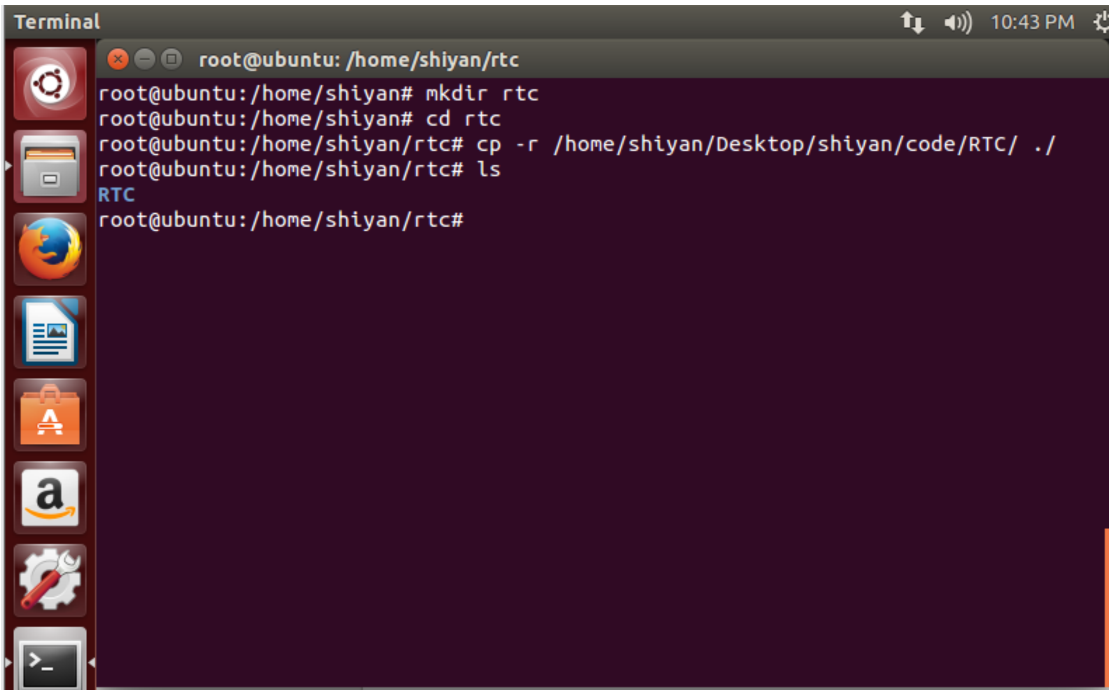
服务器：将服务器上/shiyan/2021/code 文件夹下的 RTC 目录复制到自己用户目录下。

虚拟机：将虚拟机里/Desktop/shiyan/code 文件下的 RTC 目录复制到自己用户目录下

```
# mkdir rtc
# cd rtc
```

虚拟机：`# cp -r /home/shiyan/Desktop/shiyan/code/RTC/ ./`

服务器：`# cp -r /home/shiyan/2021/code/RTC ./`



```
root@ubuntu:/home/shiyan# mkdir rtc
root@ubuntu:/home/shiyan# cd rtc
root@ubuntu:/home/shiyan/rtc# cp -r /home/shiyan/Desktop/shiyan/code/RTC/ ./
root@ubuntu:/home/shiyan/rtc# ls
RTC
root@ubuntu:/home/shiyan/rtc#
```

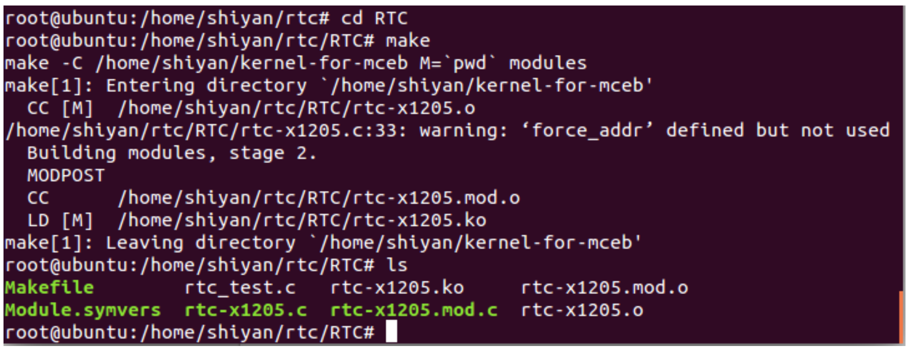
图 8-1 RTC 目录复制到自己用用户目录

步骤 2：编写用于交叉编译的 Makefile

```
KDIR:=/home/stx/ kernel-for-mceb //此处路径修改为虚拟机中存放内核文件的目录
CROSS_COMPILE = arm_v5t_le
CC = $(CROSS_COMPILE)gcc
.PHONY: modules clean
obj-m := rtc-x1205.o
modules:
        make -C $(KDIR) M=`pwd` modules
clean:
make -C $(KDIR) M=`pwd` modules clean
```

管理员权限下退到根目录执行 source /etc/profile 使环境变量生效

进入 RTC 文件夹执行 make 命令，生成 rtc-x1205.ko 文件；



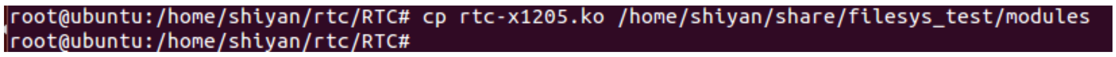
```
root@ubuntu:/home/shiyan/rtc# cd RTC
root@ubuntu:/home/shiyan/rtc/RTC# make
make -C /home/shiyan/kernel-for-mceb M=`pwd` modules
make[1]: Entering directory `/home/shiyan/kernel-for-mceb'
  CC [M]  /home/shiyan/rtc/RTC/rtc-x1205.o
/home/shiyan/rtc/RTC/rtc-x1205.c:33: warning: ‘force_addr’ defined but not used
  Building modules, stage 2.
  MODPOST
  CC      /home/shiyan/rtc/RTC/rtc-x1205.mod.o
  LD [M]  /home/shiyan/rtc/RTC/rtc-x1205.ko
make[1]: Leaving directory `/home/shiyan/kernel-for-mceb'
root@ubuntu:/home/shiyan/rtc/RTC# ls
Makefile        rtc_test.c    rtc-x1205.ko     rtc-x1205.mod.o
Module.symvers  rtc-x1205.c   rtc-x1205.mod.c  rtc-x1205.o
root@ubuntu:/home/shiyan/rtc/RTC#
```

图 8-2 生成 rtc-x1205.ko 文件

将该文件拷贝到所挂载的文件系统 filesys_test 中（根据自己的目录改路径）

虚拟机：# cp rtc-x1205.ko /home/shiyan/share/filesys_test/modules

服务器：# cp rtc-x1205.ko /home/stX/filesys_test/modules

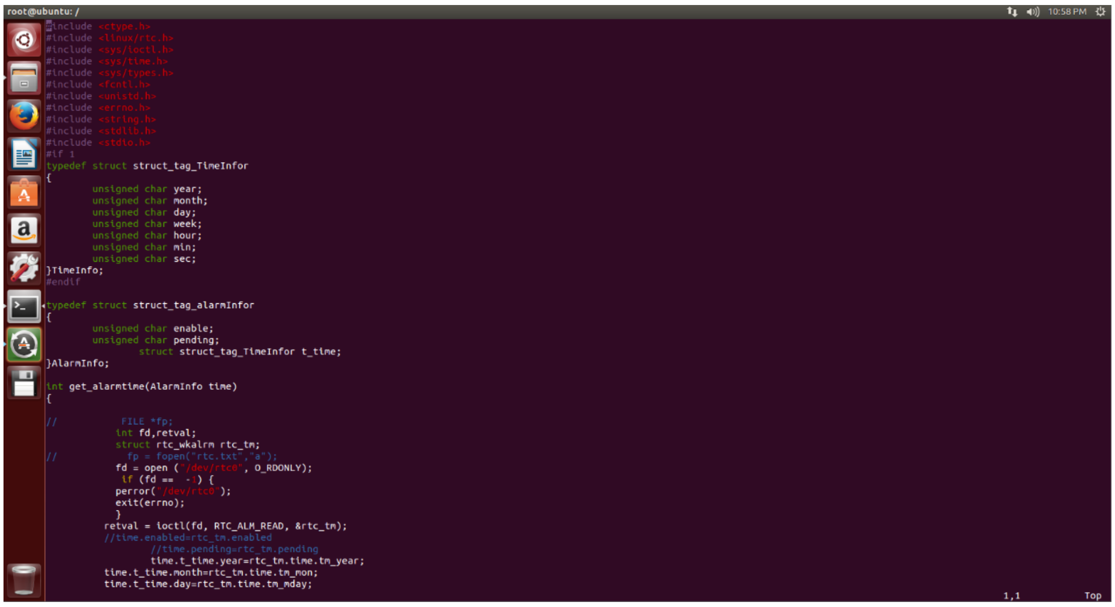


```
root@ubuntu:/home/shiyan/rtc/RTC# cp rtc-x1205.ko /home/shiyan/share/filesys_test/modules
root@ubuntu:/home/shiyan/rtc/RTC#
```

图 8-3 将 rtc-下 205.ko 拷贝到所挂载的文件系统

步骤 3：编写测试程序

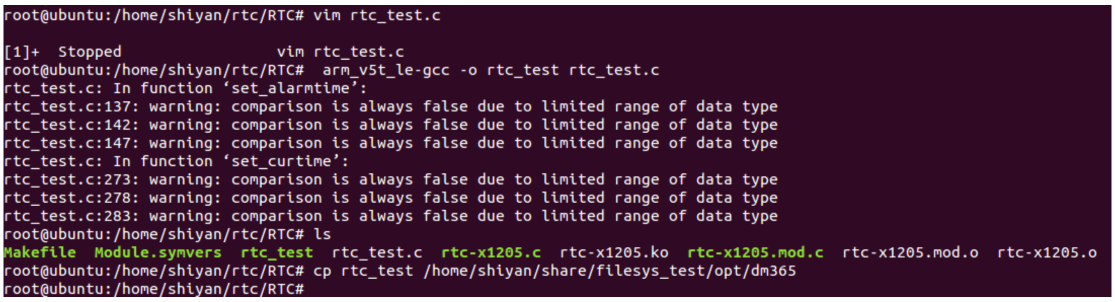
编写测试程序 rtc_test.c，将编写好的测试程序在服务器上输入命令



```
#include <ctype.h>
#include <linux/rtc.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#if 1
typedef struct struct_tag_TimeInfor
{
        unsigned char year;
        unsigned char month;
        unsigned char day;
        unsigned char week;
        unsigned char hour;
        unsigned char min;
        unsigned char sec;
}TimeInfo;
#endif

typedef struct struct_tag_alarmInfor
{
        unsigned char enable;
        unsigned char pending;
                struct struct_tag_TimeInfor t_time;
}AlarmInfo;

int get_alarmtime(AlarmInfo time)
{

//           FILE *fp;
            int fd,retval;
            struct rtc_wkalrm rtc_tm;
//            fp = fopen("rtc.txt","a");
            fd = open ("/dev/rtc0", O_RDONLY);
             if (fd ==  -1) {
            perror("/dev/rtc0");
            exit(errno);
            }
          retval = ioctl(fd, RTC_ALM_READ, &rtc_tm);
          //time.enabled=rtc_tm.enabled
                  //time.pending=rtc_tm.pending
                  time.t_time.year=rtc_tm.time.tm_year;
          time.t_time.month=rtc_tm.time.tm_mon;
          time.t_time.day=rtc_tm.time.tm_mday;
```

图 8-4 编写测试程序 rtc_test.c

# arm_v5t_le-gcc -o rtc_test rtc_test.c



```
root@ubuntu:/home/shiyan/rtc/RTC# vim rtc_test.c

[1]+  Stopped                 vim rtc_test.c
root@ubuntu:/home/shiyan/rtc/RTC#  arm_v5t_le-gcc -o rtc_test rtc_test.c
rtc_test.c: In function ‘set_alarmtime’:
rtc_test.c:137: warning: comparison is always false due to limited range of data type
rtc_test.c:142: warning: comparison is always false due to limited range of data type
rtc_test.c:147: warning: comparison is always false due to limited range of data type
rtc_test.c: In function ‘set_curtime’:
rtc_test.c:273: warning: comparison is always false due to limited range of data type
rtc_test.c:278: warning: comparison is always false due to limited range of data type
rtc_test.c:283: warning: comparison is always false due to limited range of data type
root@ubuntu:/home/shiyan/rtc/RTC# ls
Makefile  Module.symvers  rtc_test  rtc_test.c  rtc-x1205.c  rtc-x1205.ko  rtc-x1205.mod.c  rtc-x1205.mod.o  rtc-x1205.o
root@ubuntu:/home/shiyan/rtc/RTC# cp rtc_test /home/shiyan/share/filesys_test/opt/dm365
root@ubuntu:/home/shiyan/rtc/RTC#
```

图 8-5 生成可执行文件 rtc_test 并复制到挂载的文件系统

将生成可执行文件 rtc_test，将其放到文件系统 filesys_test 中：

虚拟机：# cp rtc_test /home/shiyan/share/filesys_test/opt/dm365

服务器：# cp rtc_test /home/stX/filesys_test/opt/dm365

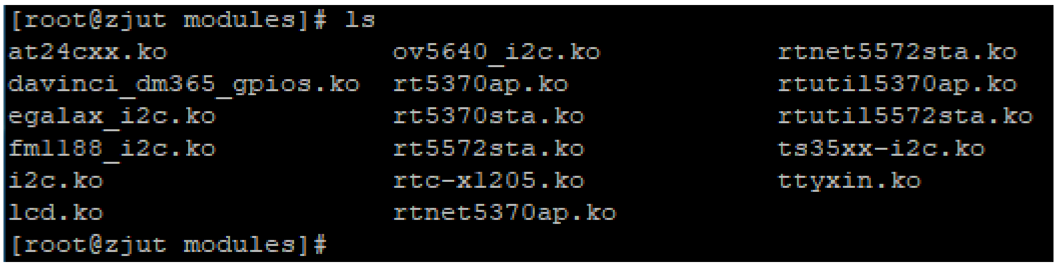
步骤 4: 挂载文件系统，设置启动参数通过 NFS 方式挂载实验箱根文件系统。

打开 putty，启动实验箱，在内核启动倒计时 5s 内按 enter 终止实验箱的

启动，输入参数挂载文件系统（参考挂载实验），然后输入 boot 引导启动。

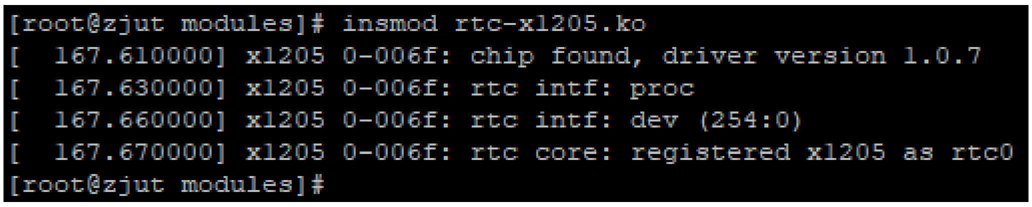
步骤 5：手动加载驱动

启动完成后，输入 root 登录板子，进入到/modules 目录，在使用 insmod rtc-x1205.ko 加载 RTC 驱动模块，如下图所示：



```
[root@zjut modules]# ls
at24cxx.ko              ov5640_i2c.ko           rtnet5572sta.ko
davinci_dm365_gpios.ko  rt5370ap.ko             rtutil5370ap.ko
egalax_i2c.ko           rt5370sta.ko            rtutil5572sta.ko
fm1188_i2c.ko           rt5572sta.ko            ts35xx-i2c.ko
i2c.ko                  rtc-x1205.ko            ttyxin.ko
lcd.ko                  rtnet5370ap.ko
[root@zjut modules]#
```

图 8-6 进入 modules 目录

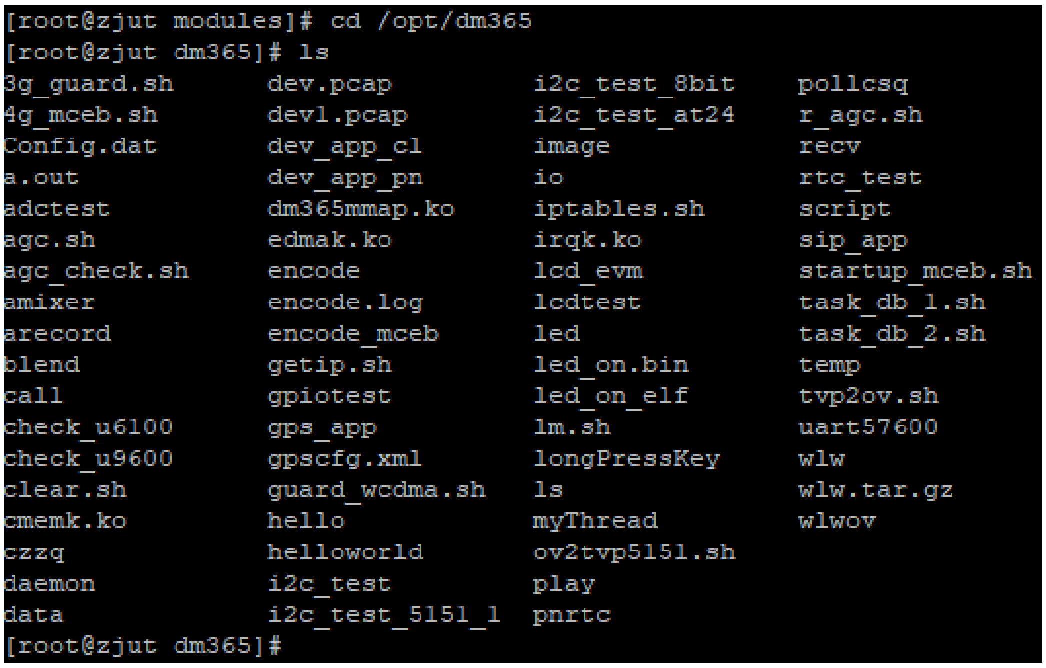


```
[root@zjut modules]# insmod rtc-x1205.ko
[  167.610000] x1205 0-006f: chip found, driver version 1.0.7
[  167.630000] x1205 0-006f: rtc intf: proc
[  167.660000] x1205 0-006f: rtc intf: dev (254:0)
[  167.670000] x1205 0-006f: rtc core: registered x1205 as rtc0
[root@zjut modules]#
```

图 8-7 加载 RTC 驱动模块

步骤 6：查找自己的测试程序

使用命令 cd/opt/dm365 进入测试程序所在目录，找到自己的测试程序，如下图所示：

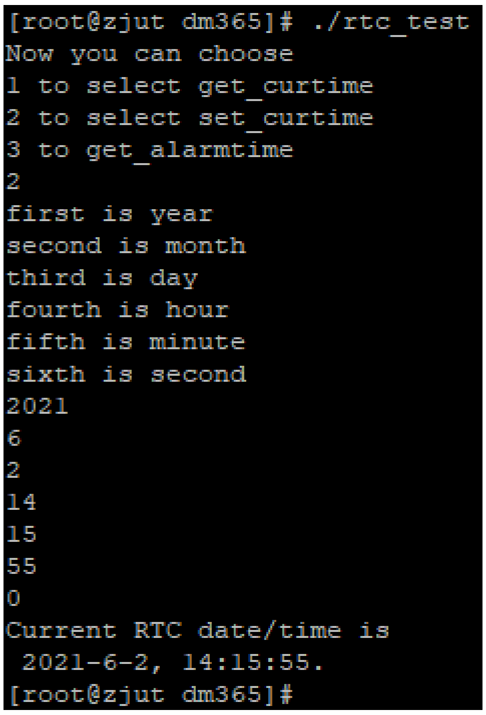


```
[root@zjut modules]# cd /opt/dm365
[root@zjut dm365]# ls
3g_guard.sh       dev.pcap          i2c_test_8bit     pollcsq
4g_mceb.sh        dev1.pcap         i2c_test_at24     r_agc.sh
Config.dat        dev_app_cl        image             recv
a.out             dev_app_pn        io                rtc_test
adctest           dm365mmap.ko      iptables.sh       script
agc.sh            edmak.ko          irqk.ko           sip_app
agc_check.sh      encode            lcd_evm           startup_mceb.sh
amixer            encode.log        lcdtest           task_db_1.sh
arecord           encode_mceb       led               task_db_2.sh
blend             getip.sh          led_on.bin        temp
call              gpiotest          led_on_elf        tvp2ov.sh
check_u6100       gps_app           lm.sh             uart57600
check_u9600       gpscfg.xml        longPressKey      wlw
clear.sh          guard_wcdma.sh    ls                wlw.tar.gz
cmemk.ko          hello             myThread          wlwov
czzq              helloworld        ov2tvp5151.sh
daemon            i2c_test          play
data              i2c_test_5151_1   pnrtc
[root@zjut dm365]#
```

图 8-8 查找自己的测试程序 rtc_test

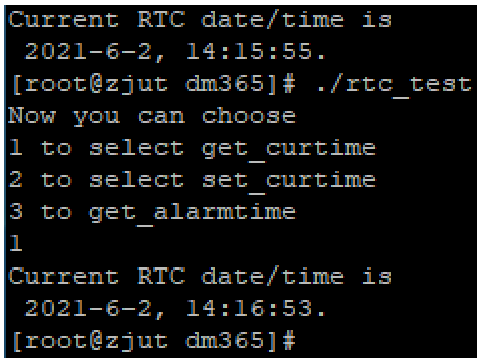
步骤 7：执行测试程序 rtc_test

执行测试程序./rtc_test，根据提示输入 1 读取当前时间，输入 2 根据提示设置时间，（先设置年，设置完后换行设置月，依次设置完，最后要多输入一个整数以示完成输入），输入 3 读取闹钟时间。



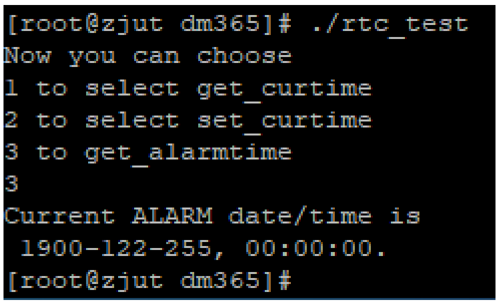
```
[root@zjut dm365]# ./rtc_test
Now you can choose
1 to select get_curtime
2 to select set_curtime
3 to get_alarmtime
2
first is year
second is month
third is day
fourth is hour
fifth is minute
sixth is second
2021
6
2
14
15
55
0
Current RTC date/time is
 2021-6-2, 14:15:55.
[root@zjut dm365]#
```

图 8-9 select set_certime



```
Current RTC date/time is
 2021-6-2, 14:15:55.
[root@zjut dm365]# ./rtc_test
Now you can choose
1 to select get_curtime
2 to select set_curtime
3 to get_alarmtime
1
Current RTC date/time is
 2021-6-2, 14:16:53.
[root@zjut dm365]#
```

图 8-10 select get_certime

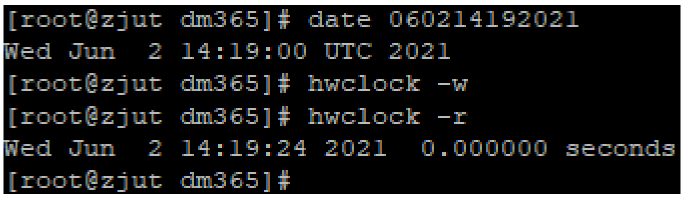


```
[root@zjut dm365]# ./rtc_test
Now you can choose
1 to select get_curtime
2 to select set_curtime
3 to get_alarmtime
3
Current ALARM date/time is
 1900-122-255, 00:00:00.
[root@zjut dm365]#
```

图 8-11 get_alarmtime

步骤 8：设置系统时间并写入硬件

可以任意设置 RTC 时间，首先使用 date xxxxxxxxxxxx（格式：月日时分年）设置系统时间，然后使用命令 hwclock -w 把系统时间写入硬件 RTC，最后使用命令 hwclock -r 读取 RTC 时间。



```
[root@zjut dm365]# date 060214192021
Wed Jun  2 14:19:00 UTC 2021
[root@zjut dm365]# hwclock -w
[root@zjut dm365]# hwclock -r
Wed Jun  2 14:19:24 2021  0.000000 seconds
[root@zjut dm365]#
```

图 8-12 设置系统时间并读入硬件

四、 心得与体会

通过对 RTC 驱动程序的编写、加载和调试，对 RTC 模块有了更加深入的了解，同时也对嵌入式驱动开发有了更为深入的了解，另一方面在驱动程序的编写也让我对嵌入式软件编程有了很大的锻炼，在前几次实验的基础上，熟悉了挂载文件系统的流程，更加熟悉了终端操作命令。

五、 附录

RTC 驱动程序源码 rtc-x1205.c：

```
/*
* An i2c driver for the Xicor/Intersil X1205 RTC
* Copyright 2004 Karen Spearel * Copyright 2005 Alessandro Zummo
*
* please send all reports to:
*  Karen Spearel <kas111@gmail.com>
*  Alessandro Zummo <a.zummo@towertech.it>
*
* based on a lot of other RTC drivers.
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation.
*/
#include <linux/module.h>
#include <linux/i2c.h>
#include <linux/bcd.h>
```

```
#include <linux/rtc.h>
#include <linux/delay.h>

#define DRV_VERSION "1.0.7"

/* Addresses to scan: none. This chip is located at
* 0x6f and uses a two bytes register addressing.
* Two bytes need to be written to read a single register,
* while most other chips just require one and take the second
* one as the data to be written. To prevent corrupting
* unknown chips, the user must explicitely set the probe parameter.
*/
static struct i2c_driver x1205_driver;
static unsigned short normal_i2c[] = { 0x6f, I2C_CLIENT_END};               //zbs tianjia
0x6f;
static unsigned short force_addr[] = {ANY_I2C_BUS, 0x6f, I2C_CLIENT_END};//zbs
//static unsigned short *force[] = {force_addr, NULL};//zbs
//static unsigned short ignore[] = {I2C_CLIENT_END};//zbs

/*static struct i2c_client_address_data addr_data =
{
.normal_i2c = normal_i2c,
.probe = ignore,
.ignore = ignore,
//.forces = forces,
};  /zbs   */

/* Insmod parameters */
I2C_CLIENT_INSMOD;

/* offsets into CCR area */
#define CCR_SEC    0
#define CCR_MIN    1
#define CCR_HOUR   2
#define CCR_MDAY   3
#define CCR_MONTH  4
#define CCR_YEAR   5
#define CCR_WDAY   6
#define CCR_Y2K    7

#define X1205_REG_SR    0x3F  /* status register */
#define X1205_REG_Y2K   0x37
#define X1205_REG_DW    0x36
#define X1205_REG_YR    0x35
```

```
#define X1205_REG_MO    0x34
#define X1205_REG_DT    0x33
#define X1205_REG_HR    0x32
#define X1205_REG_MN    0x31
#define X1205_REG_SC    0x30
#define X1205_REG_DTR   0x13
#define X1205_REG_ATR   0x12
#define X1205_REG_INT   0x11
#define X1205_REG_0     0x10
#define X1205_REG_Y2K1  0x0F
#define X1205_REG_DWA1  0x0E
#define X1205_REG_YRA1  0x0D
#define X1205_REG_MOA1  0x0C
#define X1205_REG_DTA1  0x0B
#define X1205_REG_HRA1  0x0A
#define X1205_REG_MNA1  0x09
#define X1205_REG_SCA1  0x08
#define X1205_REG_Y2K0  0x07
#define X1205_REG_DWA0  0x06
#define X1205_REG_YRA0  0x05
#define X1205_REG_MOA0  0x04
#define X1205_REG_DTA0  0x03
#define X1205_REG_HRA0  0x02
#define X1205_REG_MNA0  0x01
#define X1205_REG_SCA0  0x00

#define X1205_CCR_BASE  0x30 /* Base address ofCCR */
#define X1205_ALM0_BASE 0x00 /* Base address of ALARM0 */

#define X1205_SR_RTCF   0x01 /* Clock failure */
#define X1205_SR_WEL    0x02 /* Write Enable Latch */
#define X1205_SR_RWEL   0x04 /* Register Write Enable */

#define X1205_DTR_DTR0  0x01
#define X1205_DTR_DTR1  0x02
#define X1205_DTR_DTR2  0x04

#define X1205_HR_MIL     0x80 /* Set in ccr.hour for 24 hr mode */

/* Prototypes */
static int x1205_attach(struct i2c_adapter *adapter);
static int x1205_detach(struct i2c_client *client);
static int x1205_probe(struct i2c_adapter *adapter, int address, int kind);
static struct i2c_driver x1205_driver = {
.driver= {
   .name = "x1205",
},
.id=I2C_DRIVERID_X1205,
.attach_adapter = &x1205_attach,
.detach_client = &x1205_detach,
};

/*
* In the routines that deal directly with the x1205 hardware, we use
* rtc_time -- month 0-11, hour 0-23, yr = calendar year-epoch
* Epoch is initialized as 2000. Time is set to UTC.
*/
static int x1205_get_datetime(struct i2c_client *client, struct rtc_time *tm,
                        unsigned char reg_base)
{
      unsigned char dt_addr[2] = { 0, reg_base };
      unsigned char buf[8];

      struct i2c_msg msgs[] = {
            { client->addr, 0, 2, dt_addr }, /* setup read ptr */
            { client->addr, I2C_M_RD, 8, buf }, /* read date */
      };

      /* read date registers */
      if ((i2c_transfer(client->adapter, &msgs[0], 2)) != 2) {
      dev_err(&client->dev, "%s: read error\n", __FUNCTION__);
      return -EIO;
      }

      dev_dbg(&client->dev,
           "%s: raw read data - sec=%02x, min=%02x, hr=%02x, "
           "mday=%02x, mon=%02x, year=%02x, wday=%02x, y2k=%02x\n",
           __FUNCTION__,
           buf[0], buf[1], buf[2], buf[3],
           buf[4], buf[5], buf[6], buf[7]);
      tm->tm_sec = BCD2BIN(buf[CCR_SEC]);
      tm->tm_min = BCD2BIN(buf[CCR_MIN]);
      tm->tm_hour = BCD2BIN(buf[CCR_HOUR] & 0x3F);/* hr is 0-23 */
      tm->tm_mday = BCD2BIN(buf[CCR_MDAY]);
      tm->tm_mon = BCD2BIN(buf[CCR_MONTH]) - 1; /* mon is 0-11 */
      tm->tm_year = BCD2BIN(buf[CCR_YEAR])
                + (BCD2BIN(buf[CCR_Y2K]) * 100) - 1900;
```

```
	tm->tm_wday = buf[CCR_WDAY];

	dev_dbg(&client->dev, "%s: tm is secs=%d, mins=%d, hours=%d, "
		"mday=%d, mon=%d, year=%d, wday=%d\n",
		__FUNCTION__,
		tm->tm_sec, tm->tm_min, tm->tm_hour,
		tm->tm_mday, tm->tm_mon, tm->tm_year, tm->tm_wday);
	return 0;
}

static int x1205_get_status(struct i2c_client *client, unsigned char *sr)
{
	static unsigned char sr_addr[2] = { 0, X1205_REG_SR };

	struct i2c_msg msgs[] = {
		{ client->addr, 0, 2, sr_addr }, /* setup read ptr */
		{ client->addr, I2C_M_RD, 1, sr },
	};

	/* read status register */
	if ((i2c_transfer(client->adapter, &msgs[0], 2)) != 2) {
		dev_err(&client->dev, "%s: read error\n", __FUNCTION__);
		return -EIO;
	}

	return 0;
}

static int x1205_set_datetime(struct i2c_client *client, struct rtc_time *tm,
			int datetoo, u8 reg_base)
{
	int i, xfer;
	unsigned char buf[8];
	static const unsigned char wel[3] = { 0, X1205_REG_SR,
					X1205_SR_WEL };
	static const unsigned char rwel[3] = { 0, X1205_REG_SR,
					X1205_SR_WEL | X1205_SR_RWEL };
	static const unsigned char diswe[3] = { 0, X1205_REG_SR, 0 };

	dev_dbg(&client->dev,
	   "%s: secs=%d, mins=%d, hours=%d\n",
	   __FUNCTION__,
	   tm->tm_sec, tm->tm_min, tm->tm_hour);
```

```
        buf[CCR_SEC] = BIN2BCD(tm->tm_sec);
        buf[CCR_MIN] = BIN2BCD(tm->tm_min);

        /* set hour and 24hr bit */
        buf[CCR_HOUR] = BIN2BCD(tm->tm_hour) | X1205_HR_MIL;

        /* should we also set the date? */
        if (datetoo) {
                dev_dbg(&client->dev,
                        "%s: mday=%d, mon=%d, year=%d, wday=%d\n",
                        __FUNCTION__,
                        tm->tm_mday, tm->tm_mon, tm->tm_year, tm->tm_wday);
                buf[CCR_MDAY] = BIN2BCD(tm->tm_mday);
                /* month, 1 - 12 */
                buf[CCR_MONTH] = BIN2BCD(tm->tm_mon + 1);

                /* year, since the rtc epoch*/
                buf[CCR_YEAR] = BIN2BCD(tm->tm_year % 100);
                buf[CCR_WDAY] = tm->tm_wday & 0x07;
                buf[CCR_Y2K] = BIN2BCD(tm->tm_year / 100);
        }
        /* this sequence is required to unlock the chip */
        if ((xfer = i2c_master_send(client, wel, 3)) != 3) {
                dev_err(&client->dev, "%s: wel - %d\n", __FUNCTION__, xfer);
                return -EIO;
        }

        if ((xfer = i2c_master_send(client, rwel, 3)) != 3) {
                dev_err(&client->dev, "%s: rwel - %d\n", __FUNCTION__, xfer);
                return -EIO;
        }

        /* write register's data */
        for (i = 0; i < (datetoo ? 8 : 3); i++) {
                unsigned char rdata[3] = { 0, reg_base + i, buf[i] };

                xfer = i2c_master_send(client, rdata, 3);
                if (xfer != 3) {
                        dev_err(&client->dev,
                                "%s: xfer=%d addr=%02x, data=%02x\n",
                                __FUNCTION__,
                                xfer, rdata[1], rdata[2]);
                        return -EIO;
                }
```

```
	};

	/* disable further writes */
	if ((xfer = i2c_master_send(client, diswe, 3)) != 3) {
		dev_err(&client->dev, "%s: diswe - %d\n", __FUNCTION__, xfer);
		return -EIO
	}

	return 0;
}

static int x1205_fix_osc(struct i2c_client *client)
{
	int err;
	struct rtc_time tm;

	tm.tm_hour = tm.tm_min = tm.tm_sec = 0;

	if ((err = x1205_set_datetime(client, &tm, 0, X1205_CCR_BASE)) < 0)
		dev_err(&client->dev,
			"unable to restart the oscillator\n");

	return err;
}

static int x1205_get_dtrim(struct i2c_client *client, int *trim)
{
	unsigned char dtr; static unsigned char dtr_addr[2] = { 0, X1205_REG_DTR };
	struct i2c_msg msgs[] = {
		{ client->addr, 0, 2, dtr_addr }, /* setup read ptr */
		{ client->addr, I2C_M_RD, 1, &dtr }, /* read dtr */
	};
	/* read dtr register */
	if ((i2c_transfer(client->adapter, &msgs[0], 2)) != 2) {
		dev_err(&client->dev, "%s: read error\n", __FUNCTION__);
		return -EIO;
	}

	dev_dbg(&client->dev, "%s: raw dtr=%x\n", __FUNCTION__, dtr);

	*trim = 0;
	if (dtr & X1205_DTR_DTR0)
		*trim += 20;
	if (dtr & X1205_DTR_DTR1)
```

```
		*trim += 10;
	if (dtr & X1205_DTR_DTR2)
		*trim = -*trim;
	return 0;
}

static int x1205_get_atrim(struct i2c_client *client, int *trim)
{
	s8 atr;
	static unsigned char atr_addr[2] = { 0, X1205_REG_ATR };

	struct i2c_msg msgs[] = {
		{ client->addr, 0, 2, atr_addr }, /* setup read ptr */
		{ client->addr, I2C_M_RD, 1, &atr }, /* read atr */
	};
	/* read atr register */
	if ((i2c_transfer(client->adapter, &msgs[0], 2)) != 2) {
		dev_err(&client->dev, "%s: read error\n", __FUNCTION__);
		return -EIO;
	}

	dev_dbg(&client->dev, "%s: raw atr=%x\n", __FUNCTION__, atr);

	/* atr is a two's complement value on 6 bits,
	  * perform sign extension. The formula is
	  * Catr = (atr * 0.25pF) + 11.00pF.
	  */
	if (atr & 0x20)
		atr |= 0xC0;
	dev_dbg(&client->dev, "%s: raw atr=%x (%d)\n", __FUNCTION__, atr, atr);
	*trim = (atr * 250) + 11000;
	dev_dbg(&client->dev, "%s: real=%d\n", __FUNCTION__, *trim);
	return 0;
}

struct x1205_limit
{
	unsigned char reg, mask, min, max;
};

static int x1205_validate_client(struct i2c_client *client)
{
	int i, xfer;
```

```
/* Probe array. We will read the register at the specified
  * address and check if the given bits are zero.
  */
static const unsigned char probe_zero_pattern[] = {
       /* register, mask */
       X1205_REG_SR, 0x18,
       X1205_REG_DTR, 0xF8,
       X1205_REG_ATR, 0xC0,
       X1205_REG_INT, 0x18,
       X1205_REG_0, 0xFF,
};

static const struct x1205_limit probe_limits_pattern[] = {
       /* register, mask, min, max */
       { X1205_REG_Y2K, 0xFF, 19, 20 },
       { X1205_REG_DW, 0xFF, 0, 6},
       { X1205_REG_YR, 0xFF, 0, 99},
       { X1205_REG_MO, 0xFF, 0, 12},
       { X1205_REG_DT, 0xFF, 0, 31},
       { X1205_REG_HR, 0x7F, 0, 23},
       { X1205_REG_MN, 0xFF, 0, 59},
       { X1205_REG_SC, 0xFF, 0, 59},
       { X1205_REG_Y2K1, 0xFF, 19, 20 },
       { X1205_REG_Y2K0, 0xFF, 19, 20 },
};

/* check that registers have bits a 0 where expected */
for (i = 0; i < ARRAY_SIZE(probe_zero_pattern); i += 2) {
       unsigned char buf;
       unsigned char addr[2] = { 0, probe_zero_pattern[i] };

       struct i2c_msg msgs[2] = {
              {client->addr,0,2,addr },
              { client->addr, I2C_M_RD, 1, &buf },
       };

       if ((xfer = i2c_transfer(client->adapter, msgs, 2)) != 2)
              { dev_err(&client->adapter->dev,
              "%s: could not read register %x\n",
              __FUNCTION__, probe_zero_pattern[i]);
              return -EIO;
       }

       if ((buf& probe_zero_pattern[i+1]) != 0) {
```

```
                        dev_err(&client->adapter->dev,
                                "%s: register=%02x, zero pattern=%d, value=%x\n",
                                __FUNCTION__, probe_zero_pattern[i], i, buf);
                        return -ENODEV;
                }
        }

        /* check limits (only registers with bcd values) */
        for (i = 0; i < ARRAY_SIZE(probe_limits_pattern); i++) {
                unsigned char reg, value;
                unsigned char addr[2] = { 0, probe_limits_pattern[i].reg };
                struct i2c_msg msgs[2] = {
                        {client->addr,0,2,addr },
                        { client->addr, I2C_M_RD, 1, &reg },
                };
                if ((xfer = i2c_transfer(client->adapter, msgs, 2)) != 2) {
                        dev_err(&client->adapter->dev,
                                "%s: could not read register %x\n",
                                __FUNCTION__, probe_limits_pattern[i].reg);
                        return -EIO;
                }
                value = BCD2BIN(reg & probe_limits_pattern[i].mask);
                if (value > probe_limits_pattern[i].max ||
                value < probe_limits_pattern[i].min) {
                dev_dbg(&client->adapter->dev,
                        "%s: register=%x, lim pattern=%d, value=%d\n",
                        __FUNCTION__, probe_limits_pattern[i].reg,
                        i, value);
                return -ENODEV;
                }
        }
        return 0;
}

static int x1205_rtc_read_alarm(struct device *dev, struct rtc_wkalrm *alrm)
{
        return x1205_get_datetime(to_i2c_client(dev),
                &alrm->time, X1205_ALM0_BASE);
}

static int x1205_rtc_set_alarm(struct device *dev, struct rtc_wkalrm *alrm)
{
        return x1205_set_datetime(to_i2c_client(dev),
                &alrm->time, 1, X1205_ALM0_BASE);
```

```
}

static int x1205_rtc_read_time(struct device *dev, struct rtc_time *tm)
{
	return x1205_get_datetime(to_i2c_client(dev),
		tm, X1205_CCR_BASE);
}

static int x1205_rtc_set_time(struct device *dev, struct rtc_time *tm)
{
	return x1205_set_datetime(to_i2c_client(dev),
		tm, 1, X1205_CCR_BASE);
}

static int x1205_rtc_proc(struct device *dev, struct seq_file *seq)
{
	int err, dtrim, atrim;
	if ((err = x1205_get_dtrim(to_i2c_client(dev), &dtrim)) == 0)
		seq_printf(seq, "digital_trim\t: %d ppm\n", dtrim);
	if ((err = x1205_get_atrim(to_i2c_client(dev), &atrim)) == 0)
		seq_printf(seq, "analog_trim\t: %d.%02d pF\n",
			atrim / 1000, atrim % 1000);
	return 0;
}

static struct rtc_class_ops x1205_rtc_ops = {
	.proc = x1205_rtc_proc,
	.read_time = x1205_rtc_read_time,
	.set_time = x1205_rtc_set_time,
	.read_alarm = x1205_rtc_read_alarm,
	.set_alarm = x1205_rtc_set_alarm,
};

static ssize_t x1205_sysfs_show_atrim(struct device *dev,
			struct device_attribute *attr, char *buf)
{
	int err, atrim;
	err = x1205_get_atrim(to_i2c_client(dev), &atrim);
	if (err)
		return err;
	return sprintf(buf, "%d.%02d pF\n", atrim / 1000, atrim % 1000);
}
static DEVICE_ATTR(atrim, S_IRUGO, x1205_sysfs_show_atrim, NULL);
static ssize_t x1205_sysfs_show_dtrim(struct device *dev,
                    struct device_attribute *attr, char *buf)
{
	int err, dtrim;
	err = x1205_get_dtrim(to_i2c_client(dev), &dtrim);
	if (err)
		return err;
	return sprintf(buf, "%d ppm\n", dtrim);
}
static DEVICE_ATTR(dtrim, S_IRUGO, x1205_sysfs_show_dtrim, NULL);

static int x1205_attach(struct i2c_adapter *adapter)
{
	return i2c_probe(adapter, &addr_data, x1205_probe);
}

static int x1205_probe(struct i2c_adapter *adapter, int address, int kind)
{
	int err = 0;
	unsigned char sr;
	struct i2c_client *client;
	struct rtc_device *rtc;
	dev_dbg(&adapter->dev, "%s\n", __FUNCTION__);
	if (!i2c_check_functionality(adapter, I2C_FUNC_I2C)) {
		err = -ENODEV;
		goto exit;
	}
	if (!(client = kzalloc(sizeof(struct i2c_client), GFP_KERNEL))) {
		err = -ENOMEM;
		goto exit;
	}

	/* I2C client */
	client->addr = address;
	client->driver = &x1205_driver;
	client->adapter = adapter;

	strlcpy(client->name, x1205_driver.driver.name, I2C_NAME_SIZE);

	/* Verify the chip is really an X1205 */
	if (kind < 0) {
		if (x1205_validate_client(client) < 0) {
			err = -ENODEV;
			goto exit_kfree;
		}
	}

	/* Inform the i2c layer */
	if ((err = i2c_attach_client(client)))
		goto exit_kfree;

	dev_info(&client->dev, "chip found, driver version " DRV_VERSION "\n");

	rtc = rtc_device_register(x1205_driver.driver.name, &client->dev,
				&x1205_rtc_ops, THIS_MODULE);

	if (IS_ERR(rtc)) {
		err = PTR_ERR(rtc);
		goto exit_detach;
	}

	i2c_set_clientdata(client, rtc);

	/* Check for power failures and eventualy enable the osc */
	if ((err = x1205_get_status(client, &sr)) == 0) {
		if (sr & X1205_SR_RTCF) {
			dev_err(&client->dev,
				"power failure detected, "
				"please set the clock\n");
			udelay(50);
			x1205_fix_osc(client);
		}
	}
	else
		dev_err(&client->dev, "couldn't read status\n");

	device_create_file(&client->dev, &dev_attr_atrim);
	device_create_file(&client->dev, &dev_attr_dtrim);

	return 0;

exit_detach:
	i2c_detach_client(client);
exit_kfree:
	kfree(client);
exit:
	return err;
}
```

```
static int x1205_detach(struct i2c_client *client)
{
	int err;
	struct rtc_device *rtc = i2c_get_clientdata(client);

	if (rtc)
		rtc_device_unregister(rtc);
	if ((err = i2c_detach_client(client)))
		return err;
	kfree(client);
		return 0;
}

static int __init x1205_init(void)
{
	return i2c_add_driver(&x1205_driver);
}

static void __exit x1205_exit(void)
{
	i2c_del_driver(&x1205_driver);
}

MODULE_LICENSE("GPL");
module_init(x1205_init);
module_exit(x1205_exit);
```

测试程序 rtc_test.c

```
#include <ctype.h>
#include <linux/rtc.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#if 1
typedef struct struct_tag_TimeInfor
{
	unsigned char year;
	unsigned char month;
	unsigned char day;
	unsigned char week;
	unsigned char hour;
	unsigned char min;
	unsigned char sec;
}TimeInfo;
#endif

typedef struct struct_tag_alarmInfor
{
	unsigned char enable;
	unsigned char pending;
	struct struct_tag_TimeInfor t_time;
}AlarmInfo;

int get_alarmtime(AlarmInfo time)
{
//	FILE *fp;
	int fd,retval;
	struct rtc_wkalrm rtc_tm;
//
	fp = fopen("rtc.txt","a");
	fd = open ("/dev/rtc0", O_RDONLY);
	if (fd == -1) {
	perror("/dev/rtc0");
	exit(errno);
	}
	retval = ioctl(fd, RTC_ALM_READ, &rtc_tm);
	//time.enabled=rtc_tm.enabled
	//time.pending=rtc_tm.pending
	time.t_time.year=rtc_tm.time.tm_year;
	time.t_time.month=rtc_tm.time.tm_mon;
	time.t_time.day=rtc_tm.time.tm_mday;
	time.t_time.week=rtc_tm.time.tm_wday;
	time.t_time.hour=rtc_tm.time.tm_hour;
	time.t_time.min=rtc_tm.time.tm_min;
	time.t_time.sec=rtc_tm.time.tm_sec;
	fprintf(stdout,"CurrentALARMdate/timeis\n %d-%d-%d, %02d:%02d:%02d.\n",time.t_time.
year + 1900,
time.t_time.day,time.t_time.hour, time.t_time.min, time.t_time.sec);
	close(fd);
	return 0;
```

```
}

int set_alarmtime(AlarmInfo time)
{
       int i,fd,retval;
       int b[6];
       char *rtime[6];
       char *message[]={"first is year\n"
                     "second is month\n"
                     "third is day\n"
                     "fourth is hour\n"
                     "fifth is minute\n"
                     "sixth is second\n"};
                     struct rtc_wkalrm alarm;
                     fd = open ("/dev/rtc0", O_RDWR);
                     if (fd == -1)
                     {
                        perror("/dev/rtc0");
                        exit(errno);
                     }
                     fprintf(stdout,*message);
                     for(i=0;i<6;i++)
                     {
                        rtime[i]=(char *)malloc(sizeof(char)*10);
                        scanf("%s\n \n",rtime[i]);
                     }
                     for(i=0;i<6;i++)
                     b[i]=atoi(rtime[i]);
                     time.t_time.year=b[0]-1900;   // year
                     time.t_time.month=b[1]-1;    // month
                     time.t_time.day=b[2];       // day
                     time.t_time.hour=b[3];      // hour
                     time.t_time.min=b[4];       // minute
                     time.t_time.sec=b[5];       // second

              /*     if(time.t_time.year<2000||time.t_time.year>2099)
                     {
                        printf("Please input the correct year, the year should be in the
scope of 2000~2099!\n");
                        exit(1);
                     } */
                     if(time.t_time.month<1||time.t_time.month>12)
                     {
                        printf("Please input the correct mouth, the mouth should be in the
scope of 1~12!\n");
                       exit(1);
                    }
                    if(time.t_time.day<1||time.t_time.day>31)
                    {
                       printf("Please input the correct days, the days should be in the
scope of 1~31!\n");
                       exit(1);
                    }
                    else
if(time.t_time.month==4||time.t_time.month==6||time.t_time.month==9||time.t_time.month==1
1)
                       if(time.t_time.day<1||time.t_time.day>30)
                          {
                             printf("Please input the correct days, the days should be in
the scope of 1~30!\n");
                             exit(1);
                          }
                     }
                else if(time.t_time.month==2)
                      {
if((time.t_time.year%4==0)&&(time.t_time.year%100!=0)||(time.t_time.year%400==0))
                         { if(time.t_time.day<1||time.t_time.day>29) {
printf("Please input the correct days, the days should be in the scope of 1~29!\n");
exit(1); } } else {
if(time.t_time.day<1||time.t_time.day>28) {
printf("Please input the correct days, the days should be in the scope of 1~28!\n");
exit(1); } } }
if(time.t_time.hour<0||time.t_time.hour>23) {
printf("Please input the correct hour, the hour should be in the scope of 0~23!\n");
exit(1); }
if(time.t_time.min<0||time.t_time.min>59) {
printf("Please input the correct minute, the minute should be in the scope of 0~60!\n");
exit(1); }
if(time.t_time.sec<0||time.t_time.sec>59) {
printf("Please input the correct second, the second should be in the scope of 0~60!\n");
exit(1);


           }
             alarm.time.tm_year=time.t_time.year;
             alarm.time.tm_mon=time.t_time.month;
             alarm.time.tm_mday=time.t_time.day;
             alarm.time.tm_hour=time.t_time.hour;
             alarm.time.tm_min=time.t_time.min;
```

```
            alarm.time.tm_sec=time.t_time.sec;
            retval = ioctl(fd, RTC_ALM_SET, &alarm);
            if (retval == -1)
            {
            perror("ioctl");
            exit(errno);
            }
            close(fd);
            // system("/bin/busybox hwclock --hctosys");
            return 0;
      }

int get_curtime(TimeInfo time)
{
//   FILE *fp;
      int fd,retval;
      struct rtc_time rtc_tm;
//   fp = fopen("rtc.txt","a");
   fd = open ("/dev/rtc0", O_RDONLY);
   if (fd == -1) {
   perror("/dev/rtc0");
   exit(errno);
   }
      retval = ioctl(fd, RTC_RD_TIME, &rtc_tm);
      time.year=rtc_tm.tm_year;
      time.month=rtc_tm.tm_mon;
      time.day=rtc_tm.tm_mday;
      time.week=rtc_tm.tm_wday;
      time.hour=rtc_tm.tm_hour;
      time.min=rtc_tm.tm_min;
      time.sec=rtc_tm.tm_sec;
      fprintf(stdout, "Current RTC date/time is \n %d-%d-%d, %02d:%02d:%02d.\n",
                     time.year + 1900, time.month + 1, time.day,time.hour, time.min,
time.sec);
      close(fd);
      return 0;
}

int set_curtime(TimeInfo time)
{
           int i,fd,retval;
           int b[6];
           char *rtime[6];
           char *message[]={"first is year\n"
```

```
            "second is month\n"
            "third is day\n"
            "fourth is hour\n"
            "fifth is minute\n"
            "sixth is second\n"};
            struct rtc_time rtc_tm;
            fd = open ("/dev/rtc0", O_RDWR);
            if (fd == -1)
            {
           perror("/dev/rtc0");
           exit(errno);
            }
            fprintf(stdout,*message);
            for(i=0;i<6;i++)
            {
               rtime[i]=(char *)malloc(sizeof(char)*10);
               scanf("%s\n \n",rtime[i]);
            }
            for(i=0;i<6;i++)
                  b[i]=atoi(rtime[i]);
      time.year=b[0]-1900;    // year
      time.month=b[1]-1;    // month
      time.day=b[2];       // day
      time.hour=b[3];       // hour
      time.min=b[4];       // minute
      time.sec=b[5];       // seconde
/* if(time.year<2000||time.year>2099)
{
      printf("Please input the correct year, the year should be in the scope of 
2000~2099!\n");
      exit(1);
}
*/
if(time.month<1||time.month>12)
{
      printf("Please input the correct mouth, the mouth should be in the scope of 1~12!\n");
      exit(1);
}
if(time.day<1||time.day>31)
{
      printf("Please input the correct days, the days should be in the scope of 1~31!\n");
      exit(1);
}
else if(time.month==4||time.month==6||time.month==9||time.month==11)
```

```
{ if(time.day<1||time.day>30) {
printf("Please input the correct days, the days should be in the scope of 1~30!\n"); exit(1);
} }
else if(time.month==2) {
if((time.year%4==0)&&(time.year%100!=0)||(time.year%400==0)) {
if(time.day<1||time.day>29) {
printf("Please  input  the  correct  days,  the  days  should  be  in  the  scope  of  1~29!\n");
exit(1); } } else {
if(time.day<1||time.day>28) {
printf("Please  input  the  correct  days,  the  days  should  be  in  the  scope  of  1~28!\n");
exit(1); } } }
if(time.hour<0||time.hour>23) {
printf("Please  input  the  correct  hour,  the  hour  should  be  in  the  scope  of  0~23!\n");
exit(1); }
if(time.min<0||time.min>59) {
printf("Please input the correct minute, the minute should be in the scope of 0~60!\n");
exit(1);
} if(time.sec<0||time.sec>59) {
printf("Please input the correct second, the second should be in the scope of 0~60!\n");
exit(1); }
        rtc_tm.tm_year=time.year;
        rtc_tm.tm_mon=time.month;
        rtc_tm.tm_mday=time.day;
        rtc_tm.tm_hour=time.hour;
        rtc_tm.tm_min=time.min;
        rtc_tm.tm_sec=time.sec; retval = ioctl(fd, RTC_SET_TIME, &rtc_tm);
        if (retval == -1)
        {
                perror("ioctl");
                exit(errno);
        }
        close(fd);
        // system("/bin/busybox hwclock --hctosys");
        return 0;
      }

int main()
{
        TimeInfo p;
        AlarmInfo a;
        int choice;
        fprintf(stdout,"Now  you  can  choose\n1  to  select  get_curtime\n2  to  select
set_curtime\n3 to
get_alarmtime\n");
```

```
        //fprintf(stdout,"input any digital without zero to get time\n");
        //fprintf(stdout,"Your will get time:\n");
        scanf("%d",&choice);
        switch(choice)
        {
          case 1 :
                // set_curtime(p);
                get_curtime(p);
                break;
          case 2 :
                set_curtime(p);
                get_curtime(p);
                break;
          case 3 :
                get_alarmtime(a);
                break;
/*         case 4:
                set_curtime(p);
                get_alarmtime(a); */
           }

        // if (choice)
        // get_curtime(p);

        return 0;
}
```

实验 9：Linux 下 GPIO 驱动程序编写

一、 实验目的

1. 理解 Linux GPIO 驱动程序的结构、原理；
2. 掌握 Linux GPIO 驱动程序的编程；
3. 掌握 Linux GPIO 动态加载驱动程序模块的方法。

二、 实验内容

1. 编写 GPIO 字符设备驱动程序；
2. 编写 Makefile 文件；
3. 编写测试程序；
4. 调试 GPIO 驱动程序和测试程序。

三、 实验步骤

步骤 1：编译 GPIO 驱动（在服务器或虚拟机上进行）

创建 GPIO 文件夹，并将服务器中的 GPIO 文件夹复制过来

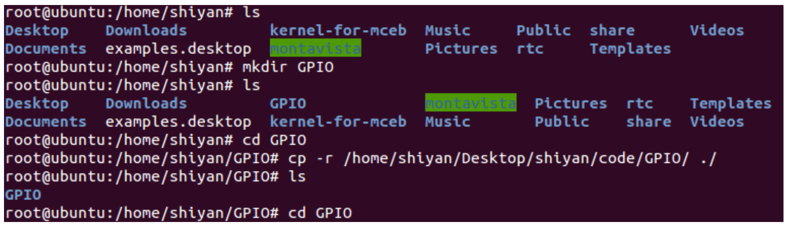
```
# mkdir GPIO
```

```
# cd GPIO
```

虚拟机：`# cp -r /home/shiyan/Desktop/shiyan/code/GPIO/ ./`

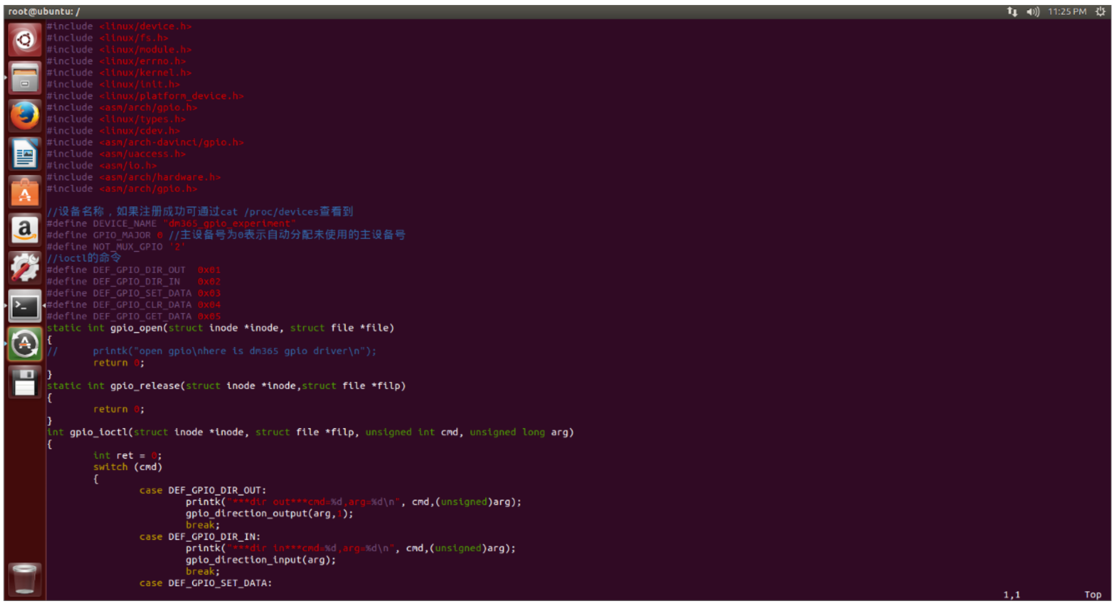
服务器：`# cp -r /home/shiyan/2021/code/GPIO/ ./`

进入 GPIO 文件夹，“cd GPIO”，GPIO 文件夹中包含三个文件，驱动程序 davinci_dm365_gpios.c，Makefile 和语言程序 gpio.c。



```
root@ubuntu:/home/shiyan# ls
Desktop    Downloads         kernel-for-mceb  Music     Public  share     Videos
Documents  examples.desktop  montavista       Pictures  rtc     Templates
root@ubuntu:/home/shiyan# mkdir GPIO
root@ubuntu:/home/shiyan# ls
Desktop    Downloads         GPIO             montavista  Pictures  rtc    Templates
Documents  examples.desktop  kernel-for-mceb  Music       Public    share  Videos
root@ubuntu:/home/shiyan# cd GPIO
root@ubuntu:/home/shiyan/GPIO# cp -r /home/shiyan/Desktop/shiyan/code/GPIO/ ./
root@ubuntu:/home/shiyan/GPIO# ls
GPIO
root@ubuntu:/home/shiyan/GPIO# cd GPIO
```

图 9-1 将 GPIO 目录复制到自己的文件夹里

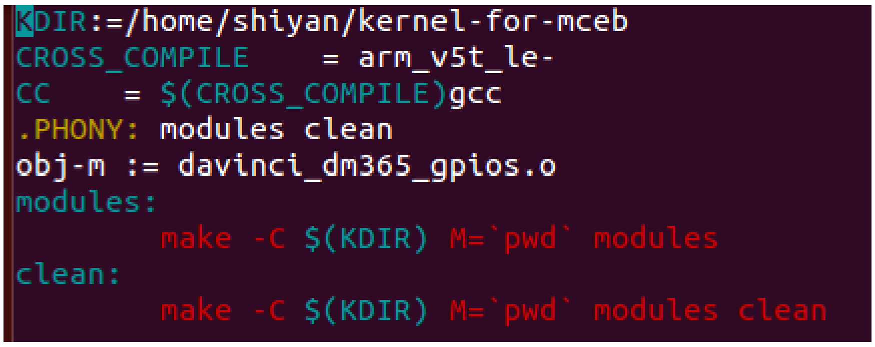


```
root@ubuntu: /
#include <linux/device.h>
#include <linux/fs.h>
#include <linux/module.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/platform_device.h>
#include <asm/arch/gpio.h>
#include <linux/types.h>
#include <linux/cdev.h>
#include <asm/arch-davinci/gpio.h>
#include <asm/uaccess.h>
#include <asm/io.h>
#include <asm/arch/hardware.h>
#include <asm/arch/gpio.h>

//设备名称，如果注册成功可通过cat /proc/devices查看到
#define DEVICE_NAME "dm365_gpio_experiment"
#define GPIO_MAJOR 0 //主设备号为0表示自动分配未使用的主设备号
#define NOT_MUX_GPIO '2'
//ioctl的命令
#define DEF_GPIO_DIR_OUT  0x01
#define DEF_GPIO_DIR_IN   0x02
#define DEF_GPIO_SET_DATA 0x03
#define DEF_GPIO_CLR_DATA 0x04
#define DEF_GPIO_GET_DATA 0x05
static int gpio_open(struct inode *inode, struct file *file)
{
//      printk("open gpio\nhere is dm365 gpio driver\n");
        return 0;
}
static int gpio_release(struct inode *inode,struct file *filp)
{
        return 0;
}
int gpio_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
        int ret = 0;
        switch (cmd)
        {
                case DEF_GPIO_DIR_OUT:
                        printk("***dir out***cmd=%d,arg=%d\n", cmd,(unsigned)arg);
                        gpio_direction_output(arg,1);
                        break;
                case DEF_GPIO_DIR_IN:
                        printk("***dir in***cmd=%d,arg=%d\n", cmd,(unsigned)arg);
                        gpio_direction_input(arg);
                        break;
                case DEF_GPIO_SET_DATA:
                                                                          1,1           Top
```

图 9-2 驱动程序源码

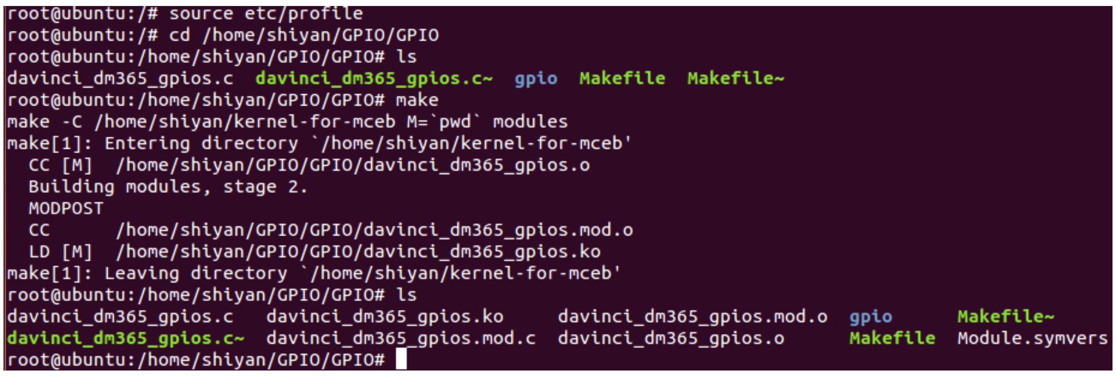
管理员权限下退到根目录执行 source /etc/profile 使环境变量生效。

进入执行“make”命令，成功后会生成 davinci_dm365_gpios.ko 等文件。



```
KDIR:=/home/shiyan/kernel-for-mceb
CROSS_COMPILE     = arm_v5t_le-
CC     = $(CROSS_COMPILE)gcc
.PHONY: modules clean
obj-m := davinci_dm365_gpios.o
modules:
        make -C $(KDIR) M=`pwd` modules
clean:
        make -C $(KDIR) M=`pwd` modules clean
```

图 9-3 Makefile



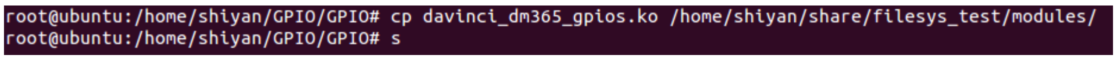
```
root@ubuntu:/# source etc/profile
root@ubuntu:/# cd /home/shiyan/GPIO/GPIO
root@ubuntu:/home/shiyan/GPIO/GPIO# ls
davinci_dm365_gpios.c  davinci_dm365_gpios.c~  gpio  Makefile  Makefile~
root@ubuntu:/home/shiyan/GPIO/GPIO# make
make -C /home/shiyan/kernel-for-mceb M=`pwd` modules
make[1]: Entering directory `/home/shiyan/kernel-for-mceb'
  CC [M]  /home/shiyan/GPIO/GPIO/davinci_dm365_gpios.o
  Building modules, stage 2.
  MODPOST
  CC      /home/shiyan/GPIO/GPIO/davinci_dm365_gpios.mod.o
  LD [M]  /home/shiyan/GPIO/GPIO/davinci_dm365_gpios.ko
make[1]: Leaving directory `/home/shiyan/kernel-for-mceb'
root@ubuntu:/home/shiyan/GPIO/GPIO# ls
davinci_dm365_gpios.c   davinci_dm365_gpios.ko     davinci_dm365_gpios.mod.o  gpio      Makefile~
davinci_dm365_gpios.c~  davinci_dm365_gpios.mod.c  davinci_dm365_gpios.o      Makefile  Module.symvers
root@ubuntu:/home/shiyan/GPIO/GPIO# 
```

图 9-4 生效环境变量后 make 生成.ko 文件

davinci_dm365_gpios.ko 文件生成成功后，将其复制到挂载的文件系统 modules 的目录下。

服务器：# cp davinci_dm365_gpios.ko /home/stX/filesys_test/modules/

虚拟机：# cp davinci_dm365_gpios.ko /home/shiyan/share/filesys_test/modules/



```
root@ubuntu:/home/shiyan/GPIO/GPIO# cp davinci_dm365_gpios.ko /home/shiyan/share/filesys_test/modules/
root@ubuntu:/home/shiyan/GPIO/GPIO# s
```

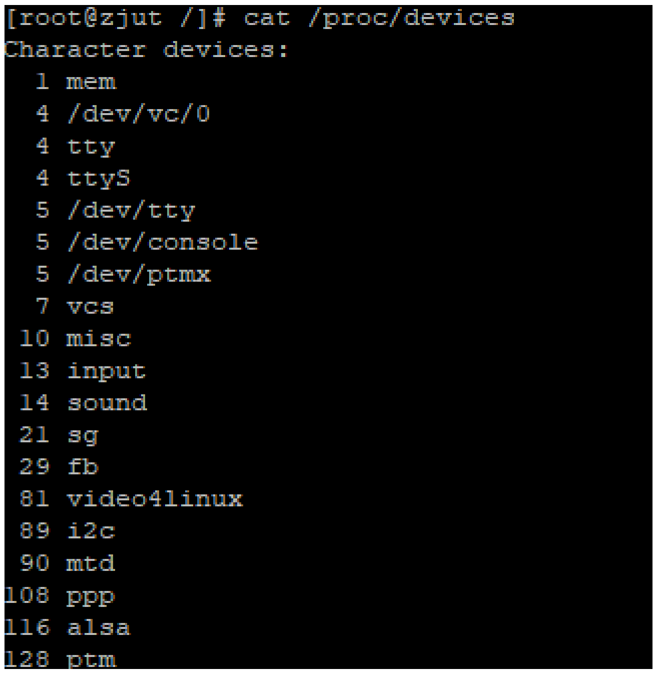
图 9-5 将驱动程序复制到文件系统

步骤 2：通过 pc 的远程登陆软件 putty 登录板子

步骤 3：动态加载模块

（1）NFS 挂载启动

正确连接好实验箱，开启电源，通过 NFS 作为启 动方式挂载虚拟机上 filesys_test 文件系统作为根文件系统进行启动。注：启动机器前要确保实验箱设备的 ip 地址和 NFS 服务器（该服务器一般和实验箱设备在同一网关下）的 ip 地址可以相互 ping 通。且实验像设备的 ip 地址是该网关下唯一的 ip 地址否则将会冲突报错。登入板子后，在驱动模块加载之前查看设备节点使用命令 cat/proc/devices。



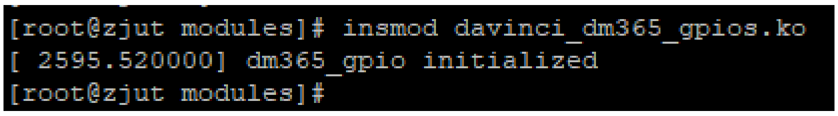
```
[root@zjut /]# cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 14 sound
 21 sg
 29 fb
 81 video4linux
 89 i2c
 90 mtd
108 ppp
116 alsa
128 ptm
```

图 9-6 cat 查看设备节点

（2）动态加载

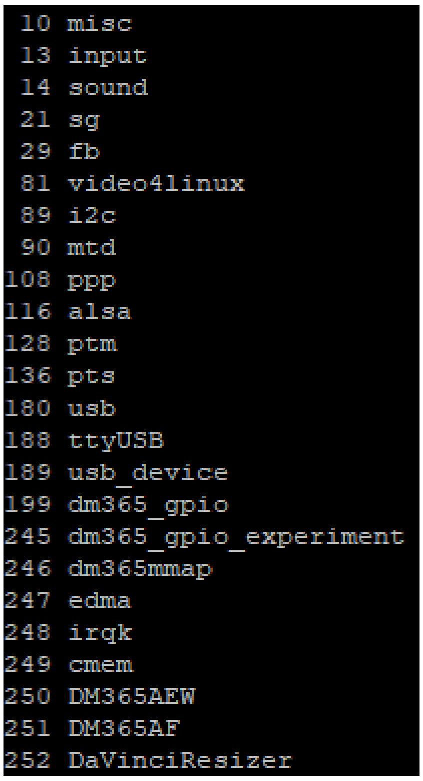
进入文件系统的 modules 目录下，查看已经加载的模块 lsmod。运行模块加载命令 insmod davinci_dm365_gpios.ko，模块成功加载，提示信息 dm365_gpio initialized。（如果加载失败，可以用 lsmod 命令查看是否已存在模块

davinci_dm365_gpios.ko，若已存在，可用 rmmod davinci_dm365_gpios.ko 先卸载，再依照上述方式加载查看结果）。



```
[root@zjut modules]# insmod davinci_dm365_gpios.ko
[ 2595.520000] dm365_gpio initialized
[root@zjut modules]#
```

图 9-7 dm365_gpio 模块加载成功



```
 10 misc
 13 input
 14 sound
 21 sg
 29 fb
 81 video4linux
 89 i2c
 90 mtd
108 ppp
116 alsa
128 ptm
136 pts
180 usb
188 ttyUSB
189 usb_device
199 dm365_gpio
245 dm365_gpio_experiment
246 dm365mmap
247 edma
248 irqk
249 cmem
250 DM365AEW
251 DM365AF
252 DaVinciResizer
```

图 9-8 重新查看已加载模块

步骤 4：手动创建设备节点（以下操作在板子上进行）

当使用 insmod 命令成功加载 davinci_dm365_gpios.ko 驱动模块后，再次通通过命令 cat /proc/devices 查看已经注册的设备和设备号，可以发现 dm365_gpio_experiment 驱动已经加载完成。

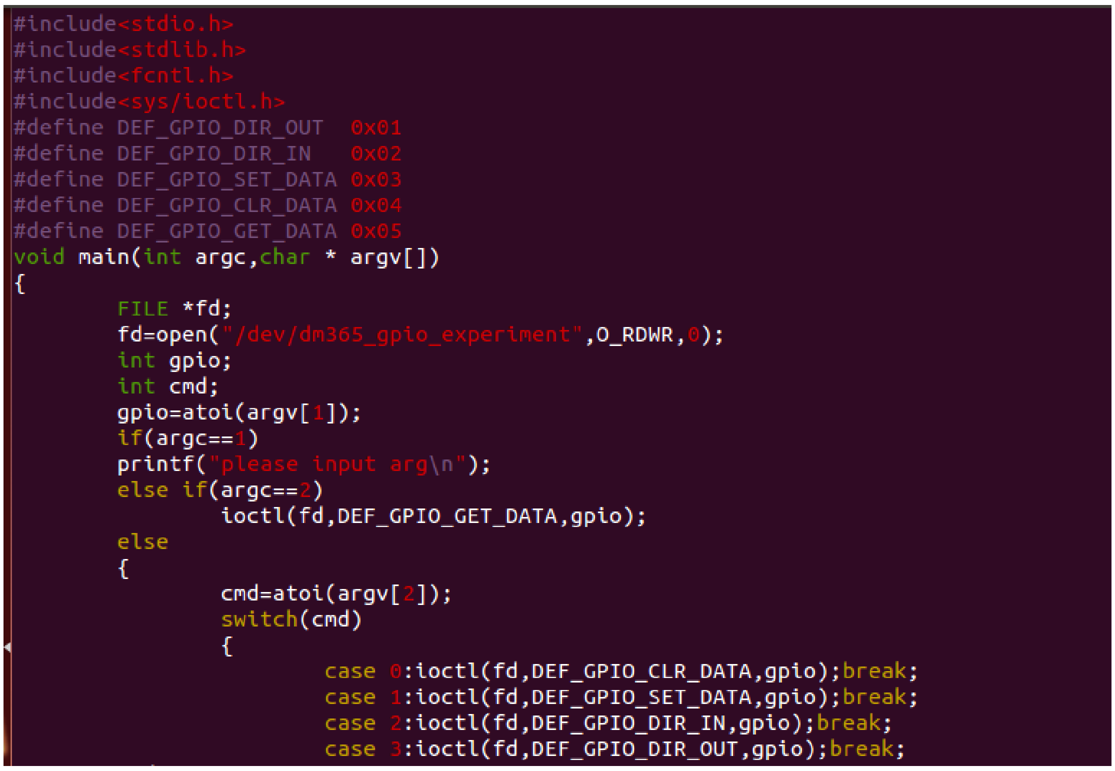
驱动模块的设备节点创建有俩种方式手动创建和自动创建，gpio 驱动程序为手动创建设备节点。

```
mknod /dev/dm365_gpio_experiment c 245 0
```

（其中，dm365_gpio_experiment 是设备文件名，c 代表字符设备，245 代表上图中 dm365_gpio_experiment 对应的主设备号，0 表示次设备号，主设备号是 0 因此这个设备号是随机分配未使用的设备号，因此每次加载模块产生的设备号可能是不一样的。）

步骤 5：编写测试程序（在虚拟机上进行），并进行调试（在实验箱上进行）

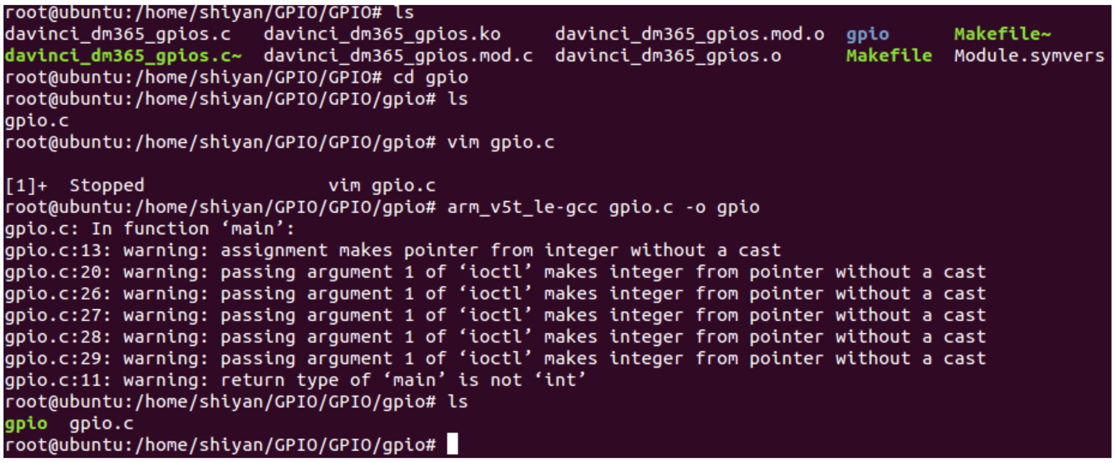
测试程序在文件下 GPIO/gpio/gpio.c 下，gpio.c.就是对应的测试文件查看实验测试程序。



```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/ioctl.h>
#define DEF_GPIO_DIR_OUT  0x01
#define DEF_GPIO_DIR_IN   0x02
#define DEF_GPIO_SET_DATA 0x03
#define DEF_GPIO_CLR_DATA 0x04
#define DEF_GPIO_GET_DATA 0x05
void main(int argc,char * argv[])
{
        FILE *fd;
        fd=open("/dev/dm365_gpio_experiment",O_RDWR,0);
        int gpio;
        int cmd;
        gpio=atoi(argv[1]);
        if(argc==1)
        printf("please input arg\n");
        else if(argc==2)
                ioctl(fd,DEF_GPIO_GET_DATA,gpio);
        else
        {
                cmd=atoi(argv[2]);
                switch(cmd)
                {
                        case 0:ioctl(fd,DEF_GPIO_CLR_DATA,gpio);break;
                        case 1:ioctl(fd,DEF_GPIO_SET_DATA,gpio);break;
                        case 2:ioctl(fd,DEF_GPIO_DIR_IN,gpio);break;
                        case 3:ioctl(fd,DEF_GPIO_DIR_OUT,gpio);break;
```

图 9-9 测试程序 gpio.c

在服务器编写测试程序的各函数后，接下来是使用交叉编译工具编译测试程序，并将编译后生成的可执行文件挂载到实验箱的板子上运行调试。

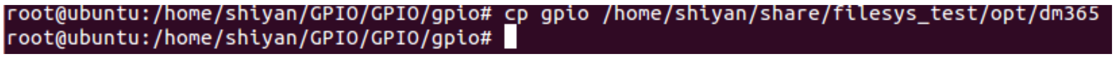


```
root@ubuntu:/home/shiyan/GPIO/GPIO# ls
davinci_dm365_gpios.c   davinci_dm365_gpios.ko     davinci_dm365_gpios.mod.o  gpio      Makefile~
davinci_dm365_gpios.c~  davinci_dm365_gpios.mod.c  davinci_dm365_gpios.o      Makefile  Module.symvers
root@ubuntu:/home/shiyan/GPIO/GPIO# cd gpio
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# ls
gpio.c
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# vim gpio.c

[1]+  Stopped                 vim gpio.c
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# arm_v5t_le-gcc gpio.c -o gpio
gpio.c: In function ‘main’:
gpio.c:13: warning: assignment makes pointer from integer without a cast
gpio.c:20: warning: passing argument 1 of ‘ioctl’ makes integer from pointer without a cast
gpio.c:26: warning: passing argument 1 of ‘ioctl’ makes integer from pointer without a cast
gpio.c:27: warning: passing argument 1 of ‘ioctl’ makes integer from pointer without a cast
gpio.c:28: warning: passing argument 1 of ‘ioctl’ makes integer from pointer without a cast
gpio.c:29: warning: passing argument 1 of ‘ioctl’ makes integer from pointer without a cast
gpio.c:11: warning: return type of ‘main’ is not ‘int’
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# ls
gpio  gpio.c
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# 
```

图 9-10 交叉编译生成可执行文件 gpio

交叉编译生成可执行文件 gpio。编译成功后，可看见 gpio 文件，将可执行文 gpi 复制到板子挂载的文件系统/home/shiyan/share/filesys_test/opt/dm365。交叉编

译生成可执行文件 gpio。 编译成功后，将可执行文件 gpio 复制到 /home/shiyan/share/filesys_test/opt/dm365 文件夹下，此时对实验箱进行操作进入 /opt/dm365 目录下执行测试代码。



```
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# cp gpio /home/shiyan/share/filesys_test/opt/dm365
root@ubuntu:/home/shiyan/GPIO/GPIO/gpio# 
```

图 9-11 将 gpio 可执行文件复制进入 opt 的 dm365 目录下

虚拟机：# cp gpio /home/shiyan/share/filesys_test/opt/dm365

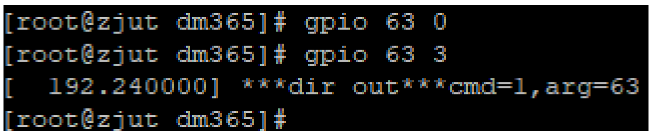
服务器：# cp gpio /home/stX/file sys_test/opt/dm365

实验箱：# cd /opt/dm365

执行如下命令 gpio 63 0/3。观察实验箱上液晶屏暗亮。

# gpio 63 0 （实验箱的板子上运行）lcd 背光打开

# gpio 63 3 （实验箱的板子上运行） lcd 背光关闭



```
[root@zjut dm365]# gpio 63 0
[root@zjut dm365]# gpio 63 3
[   192.240000] ***dir out***cmd=1,arg=63
[root@zjut dm365]#
```

图 9-12 输入控制 LCD 背光指令

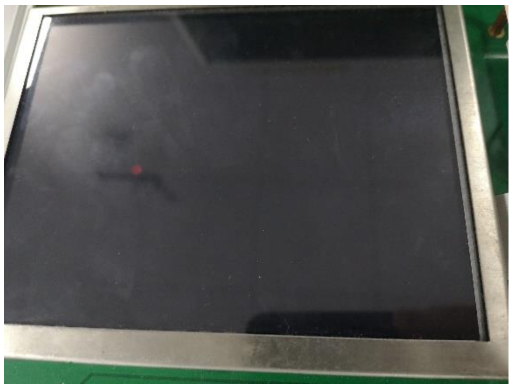


图 9-13 LCD 灭

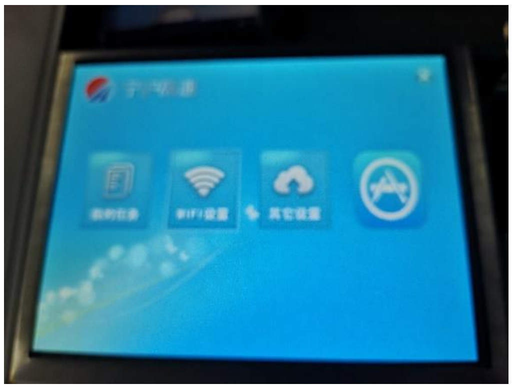


图 9-14 LCD 亮

四、 心得与体会

GPIO 口是嵌入式设备的重要模块，在嵌入式开发中的意义重大，通过此次驱动实验，对驱动程序的开发有了直观的认识，但是同样也发现，驱动程序的编写量非常大，仅仅靠这一两次实验是远远不够的，课后还是需要我们认真分析代码尝试自己写代码。

五、 附录

GPIO 驱动主要代码

```
#include <linux/device.h>
#include <linux/fs.h>
#include <linux/module.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/platform_device.h>
#include <asm/arch/gpio.h>
#include <linux/types.h>
#include <linux/cdev.h>
#include <asm/arch-davinci/gpio.h>
#include <asm/uaccess.h>
include <asm/io.h>
#include <asm/arch/hardware.h>
#include <asm/arch/gpio.h>

//设备名称，如果注册成功可通过 cat /proc/devices 查看到
#define DEVICE_NAME "dm365_gpio_experiment"
#define GPIO_MAJOR 0 //主设备号为 0 表示自动分配未使用的主设备号
#define NOT_MUX_DATA 0x03
#define DEF_GPIO_CLR_DATA 0x04
#define DEF_GPIO_GET_DATA 0x05
static int gpio_open(struct inode *inode, struct file *file)
{
//      printk("open gpio\nhere is dm365 gpio driver\n");
       return 0;
}
static int gpio_release(struct inode *inode,struct file *filp)
{
      return 0;
}
int gpio_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
int ret = 0;
switch (cmd)
{
case DEF_GPIO_DIR_OUT:
   printk("***dir out***cmd=%d,arg=%d\n",cmd,(unsigned)arg);
   gpio_direction_output(arg,1);
   break;
case DEF_GPIO_DIR_IN:
   printk("***dir in***cmd=%d,arg=%d\n", cmd,(unsigned)arg);
   gpio_direction_input(arg);
   break;
case DEF_GPIO_SET_DATA:
case DEF_GPIO_CLR_DATA:
//   printk("data=%d\n", cmd);
   if (cmd == DEF_GPIO_SET_DATA)
      gpio_set_value(arg, 1);
   else
      gpio_set_value(arg, 0);
//   break;
case DEF_GPIO_GET_DATA:
   ret = gpio_get_value(arg);
//   printk("gpio%d = %d\n",arg,ret);
   break;
   }
   return ret;
}

static struct file_operations gpio_fops = {
   .owner=THIS_MODULE,
   .open=gpio_open,
   .release=gpio_release, .ioctl=gpio_ioctl
};
static int __init davinci_dm365_gpio_init (void)
{
   int ret;
   ret=register_chrdev(GPIO_MAJOR, DEVICE, &gpio_fops);
   if(ret < 0)
   {
      printk("dm365_gpio register falid!\n");
      return ret;
```

```
    }
    printk ("dm365_gpio initialized\n");
    return ret;
}
static void __exit davinci_dm365_gpio_exit(void)
{
       unregister_chrdev(GPIO_MAJOR, DEVICE_NAME);
       printk("dm365_gpio exit\n");
}
module_init(davinci_dm365_gpio_init);
module_exit(davinci_dm365_gpio_exit);
MODULE_AUTHOR("jingwei,Song <Punuo Ltd.>");
MODULE_DESCRIPTION("Davinci DM365 gpio driver");
MODULE_LICENSE("GPL");
```