

实验报告

课程：嵌入式系统A

第八次实验

**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23日**

实验7：I2C驱动实验

1 实验目的

1. 熟悉I2C协议的原理；
2. 熟悉Linux下I2C的驱动构架；
3. 熟悉Linux驱动程序的编写；
4. 熟悉Linux下模块的加载等。

2 实验内容

1. 编写I2C协议的驱动程序；
2. 编写Makefile；
3. 以模块方式加载驱动程序；
4. 编写I2C测试程序。

3 实验步骤

**步骤1：**硬件连接

查看串口号，通过putty软件使用串口通信方式连接实验箱。



图8-1 查看实验箱端口号

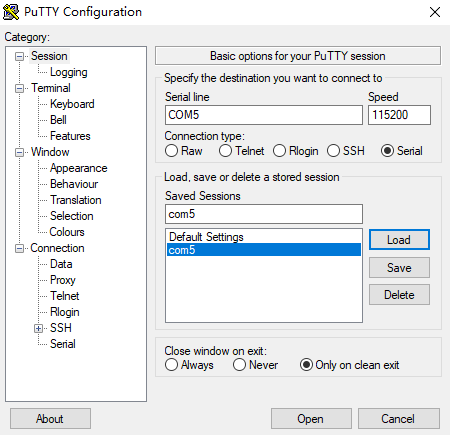


图8-2 打开putty连接

输入启动参数启动内核

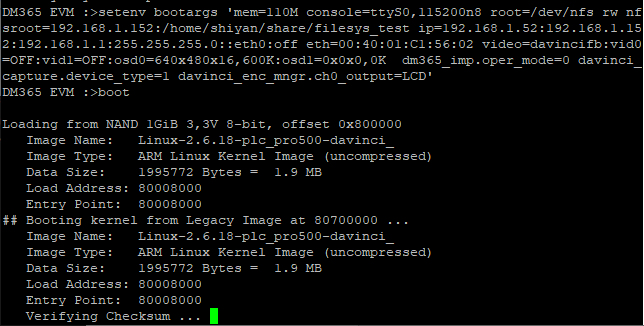


图8-3 输入启动参数启动

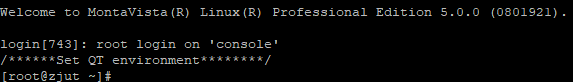


图8-4 实验箱启动成功并且用root登录

**步骤2：**编译I2C驱动

进入学生目录，mkdir I2C，创建I2C驱动文件夹，编写I2C驱动程序。

设备驱动源码参考文件夹I2C驱动实验/I2C/i2c.c。

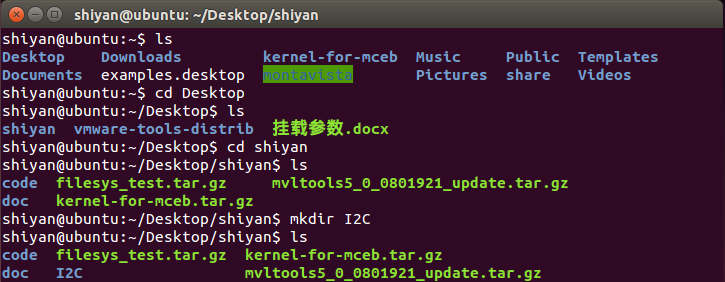


图8-5 创建I2C驱动文件夹

测试程序源码参考文件夹I2C驱动实验/I2C/i2c\_test.c

将code文件夹下的I2C程序复制到自己创建的目录下，输入命令：

|  |
| --- |
| 虚拟机：cp /home/shiyan/Desktop/shiyan/code/I2C/I2C\_CODE/\* I2C/  服务器：cp /home/shiyan/2021/code/I2C/I2C\_CODE/\* I2C/ |

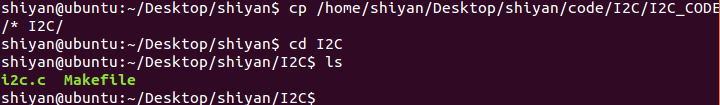


图8-6 将code文件夹下的I2C程序复制到自己创建的目录下

进入自己创建的文件夹cd I2C

编写驱动程序编译成模块所需要的Makefile文件，执行vim Makefile

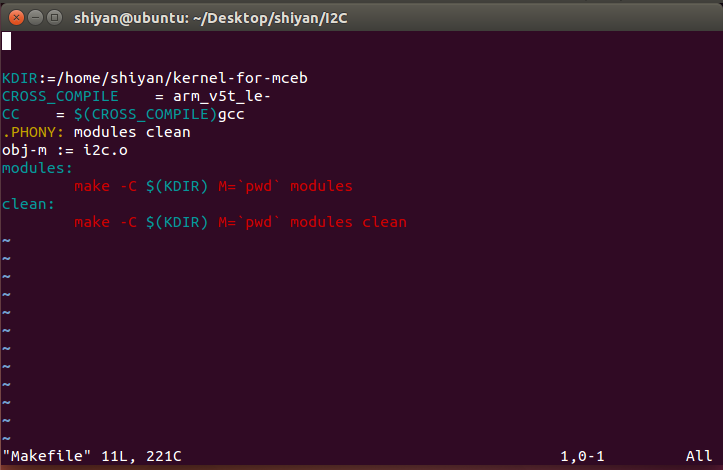


图8-7 Makefile文件

Makefile文件如下：

|  |
| --- |
| KDIR:=/home/shiyan/kernel-for-mceb  CROSS\_COMPILE = arm\_v5t\_le-  CC = $(CROSS\_COMPILE)gcc  .PHONY: modules clean  obj-m := i2c.o  modules:  make -C $(KDIR) M=`pwd` modules  clean:  make -C $(KDIR) M=`pwd` modules clean |

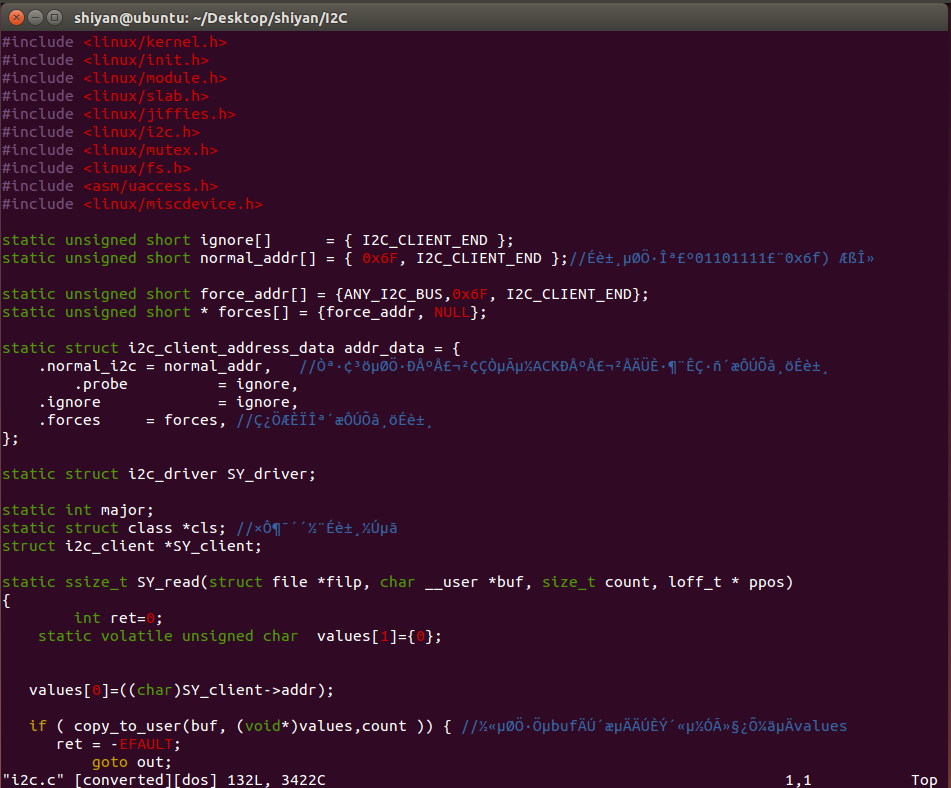


图8-8 i2c.c

i2c驱动如下：

|  |
| --- |
| #include <linux/kernel.h>  #include <linux/init.h>  #include <linux/module.h>  #include <linux/slab.h>  #include <linux/jiffies.h>  #include <linux/i2c.h>  #include <linux/mutex.h>  #include <linux/fs.h>  #include <asm/uaccess.h>  #include <linux/miscdevice.h>  static unsigned short ignore[] = { I2C\_CLIENT\_END };  static unsigned short normal\_addr[] = { 0x6F, I2C\_CLIENT\_END };//设备地址为：01101111（0x6f) 七位  static unsigned short force\_addr[] = {ANY\_I2C\_BUS,0x6F, I2C\_CLIENT\_END};  static unsigned short \* forces[] = {force\_addr, NULL};  static struct i2c\_client\_address\_data addr\_data = {  .normal\_i2c = normal\_addr, //要发出地址信号，并且得到ACK信号，才能确定是否存在这个设备  .probe = ignore,  .ignore = ignore,  .forces = forces, //强制认为存在这个设备  };  static struct i2c\_driver SY\_driver;  static int major;  static struct class \*cls; //自动创建设备节点  struct i2c\_client \*SY\_client;  static ssize\_t SY\_read(struct file \*filp, char \_\_user \*buf, size\_t count, loff\_t \* ppos)  {  int ret=0;  static volatile unsigned char values[1]={0};    values[0]=((char)SY\_client->addr);    if ( copy\_to\_user(buf, (void\*)values,count )) { //将地址值buf内存的内容传到用户空间的values  ret = -EFAULT;  goto out;  }  out:  return ret;  }  //定义字符设备结构体  static struct file\_operations SY\_fops = {  .owner = THIS\_MODULE,  .read = SY\_read,  };  static int SY\_detect(struct i2c\_adapter \*adapter, int address, int kind)  {    printk("SY\_detect\n");  //构建一个i2c\_client结构体；收费数据主要靠它，里面有 .address .adapter .driver  SY\_client = kzalloc(sizeof(struct i2c\_client), GFP\_KERNEL);  SY\_client->addr = address;  SY\_client->adapter = adapter;  SY\_client->driver = &SY\_driver;  strcpy(SY\_client->name, "SY");  i2c\_attach\_client(SY\_client);//等要卸载驱动时，会调用 I2C\_detach  printk("SY\_probe with name = %s, addr = 0x%x\n", SY\_client->name, SY\_client->addr);    major = register\_chrdev(0, "SY", &SY\_fops);//申请字符设备主设备号  cls = class\_create(THIS\_MODULE, "SY");//创建一个类 ，然后在类下面创建一个设备  class\_device\_create(cls, NULL, MKDEV(major, 0), NULL, "SY");  return 0;  }  static int SY\_attach(struct i2c\_adapter \*adapter)  {  return i2c\_probe(adapter, &addr\_data, SY\_detect);  }  static int SY\_detach(struct i2c\_client \*client)  {  printk("SY\_detach\n");  class\_device\_destroy(cls, MKDEV(major, 0));  class\_destroy(cls);  unregister\_chrdev(major, "SY");  i2c\_detach\_client(client);//client 结构体  kfree(i2c\_get\_clientdata(client));//释放client的内存  return 0;  }  //定义i2c\_driver结构体  static struct i2c\_driver SY\_driver = {  .driver = {  .name = "SY",  },  .attach\_adapter = SY\_attach,  .detach\_client = SY\_detach,  };  static int SY\_init(void)  {  printk("SY\_init\n");  i2c\_add\_driver(&SY\_driver);//注册i2c驱动  return 0;  }  static void SY\_exit(void)  {  printk("SY\_exit\n");  i2c\_del\_driver(&SY\_driver);  }  module\_init(SY\_init);  module\_exit(SY\_exit);  MODULE\_LICENSE("GPL"); |

执行make命令，生成i2c.ko等文件

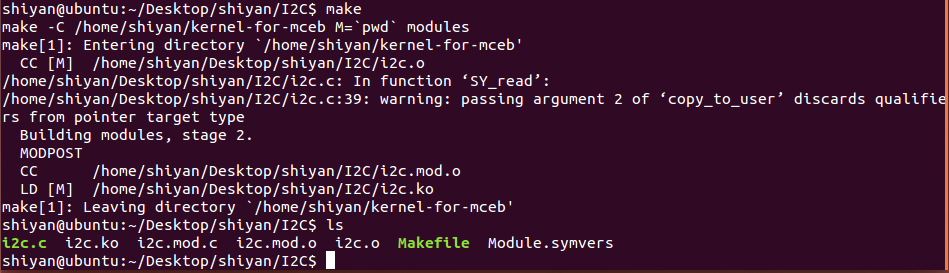


图8-9 make生成i2c.ko等文件

i2c.ko驱动文件生成成功后，将其复制到挂载的文件系统modules的目录下。

|  |
| --- |
| 虚拟机：sudo cp i2c.ko /home/shiyan/share/filesys\_test/modules  服务器：sudo cp i2c.ko /home/stX/filesys\_test/modules |

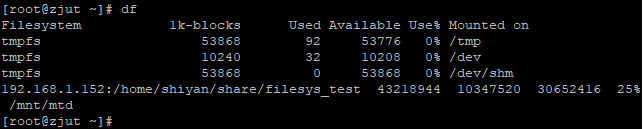


图8-10 成功挂载文件系统

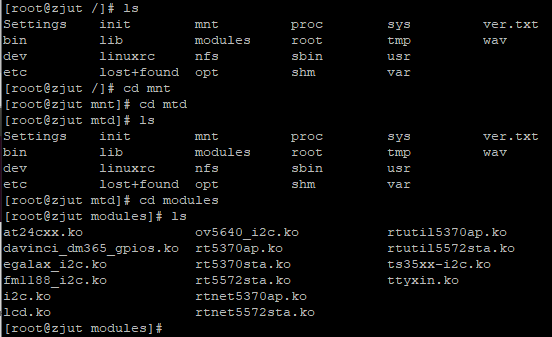


图8-11 在实验箱中查看已经有了相应文件

**步骤3：**加载驱动

1. 在驱动加载之前查看已经加载的驱动模块，使用命令lsmod。

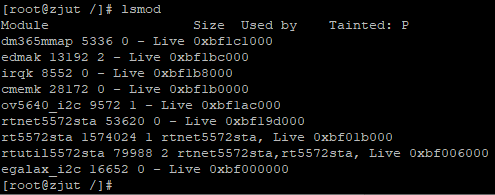


图8-12 lsmod查看已加载的驱动模块

1. 执行cd /sys/bus/i2c，进入目录查看当前加载的设备地址和设备名字。所有的I2C设备都在sysfs文件系统中显示。在当前目录文件夹下的devices下，是当前挂载在I2C的设备地址。在drivers目录下是挂载I2C上的设备驱文件。

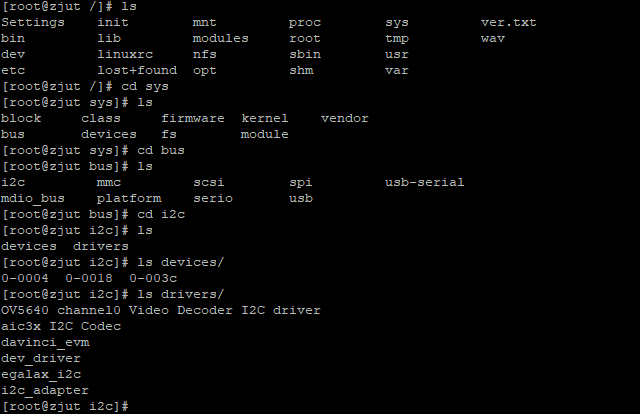


图8-13 查看当前加载的设备地址和设备名

1. 执行insmod /modules/i2c.ko。在驱动加载成功后会打印出设备的I2C注册地址。具体添加设备的地址查看数据手册，每一个设备都会有一个固定地址。添加设备地址为0x6f，转化成二进制为一个8位的数据。I2C协议地址为7位。

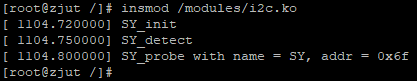


图8-14 手动加载驱动模块

1. 驱动加载成功之后，查看devices和drivers目录，会增加一个0-006f文件。

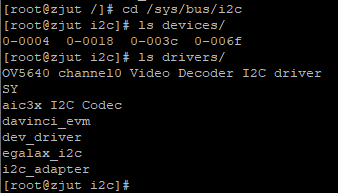


图8-15 设备SY注册成功

1. 执行cat /proc/device，显示加载的I2C设备驱动程序创建了一个主设备号为245名为SY的设备节点。

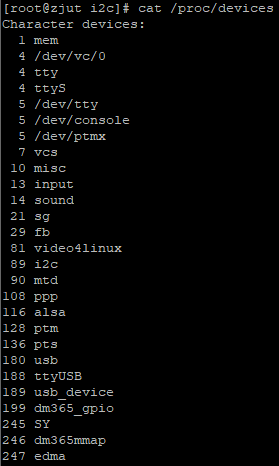


图8-16 创建了设备号245名为SY的设备节点

**步骤4：**编写测试程序，并进行调试

测试程序在文件夹I2C驱动实验/I2C/i2c\_test下，i2c\_test.c.就是对应的测试文件。用来查看设备的地址。

i2c\_test.c测试代码：

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <fcntl.h>  int main(int argc, char \*\*argv)  {  int fd,ret;  unsigned char values[0];  fd = open("/dev/SY", O\_RDWR);  if (fd < 0)  {  printf("can't open /dev/SY\n");  return -1;  }  ret= read(fd,values,sizeof(unsigned char));  if (ret >= 0){  printf("reading data is OK \n");  }  else{  printf("read data is fialed \n",ret);  }  printf("SY address = 0x%x\n",values[0]);  return 0;  } |

重置生效环境变量，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上运行调试。

$ arm\_v5t\_le-gcc i2c\_test.c -o i2c\_test

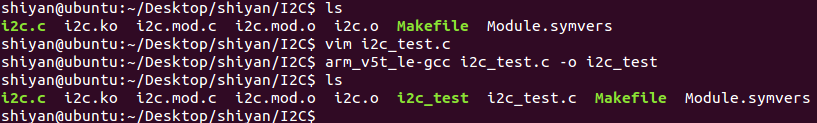


图8-17 生成i2c\_test可执行文件

将交叉编译生成的i2c\_test文件拷贝到挂载的文件系统目录下



putty实验箱窗口执行i2c\_test，读出当前设备地址。

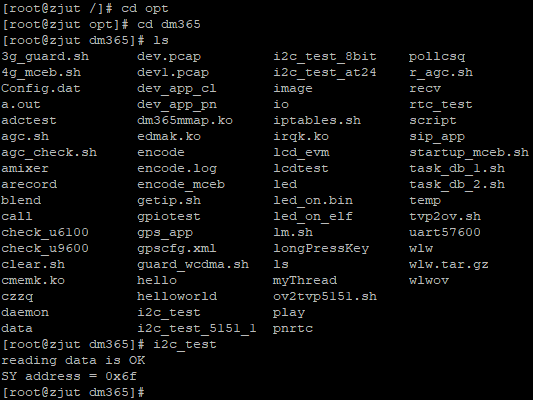


图8-18 测试执行程序

4 心得与体会

这是嵌入式系统的最后一次实验，通过这次实验，对I2C驱动程序的编写有更为深入的了解；I2C为常见的总线接口，后续学习和工作中很有可能会经常用到，所以我们还是要继续深入了解熟悉I2C相关驱动程序，对于其驱动的理解将为我们后续从事嵌入式开发打下基础。