实验 8 嵌入式 LINUX 下 LED 点亮实验

实验目的

- 掌握嵌入式 LINUX 硬件设备编程方法，点亮 LED 灯。

实验设备

- 硬件：PC 机+Tiny6410 开发板
- 软件：WINXP+VMWARE 虚拟机+UBUNTU10.04-32bit+arm-linux-gcc

实验内容

- 1.编写程序 embed_led.c，用来点亮核心板上的 LED 灯。
- 2.交叉编译该程序，把生成的可执行程序下载到开发板执行，观看运行结果。

实验原理

1. Open 函数

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char* pathname, int oflag, …/*mode_t mode */)
```

2.close 函数

#include <unistd.h>

int close (int *filedes*)

3.ioctl 函数：设备控制接口函数 ioctl()。

头文件：#include<unistd.h>

功 能: 控制 I/O 设备 ，提供了一种获得设备信息或设备控制参数的手段。

原型：int ioctl(int fd, ind cmd, …);

返回值：成功为 0，出错为-1

其中 fd 就是用户程序打开设备时使用 open 函数返回的文件标示符，cmd 就是用户程序对设备的控制命令，至于后面的省略号，那是一些补充参数，一般最多一个，有或没有是和 cmd 的意义相关的。

ioctl 函数是文件结构中的一个属性分量，就是说如果你的驱动程序提供了对 ioctl 的支持，用户就可以在用户程序中使用 ioctl 函数控制设备的 I/O 通道。

使用 ioctl 来实现控制的功能。要记住，用户程序所作的只是通过命令码告诉驱动程序它想做什么，至于怎么解释这些命令和怎么实现这些命令，这都是驱动程序要做的事情。在驱动程序中实现的 ioctl 函数体内，实际上是有一个 switch{case}结构，每一个 case 对应一个命令码，做出一些相应的操作。怎么实现这些 操作，这是每一个程序员自己的事情，因为设备都是特定的，这里也没法说。关键在于怎么样组织命令码，因为在 ioctl

中命令码是唯一联系用户程序命令和驱 动程序支持的途径。

实验步骤

1) 在虚拟 UNBUNT 下编辑源程序 embed_led.c

2) 编写 Makefile 文件，执行 make，生成可执行文件 embed_led（或者在虚拟 UNBUNT 下直接执行交叉编译 arm-linux-gcc，生成可执行文件 embed_led）

3) 把 embed_led 拷贝到 windows 下。在虚拟 LINUX 这个文件所在目录执行如下命令：

cp embed_led /mnt/hgfs/share

指令执行成功后，会在 WINDOWS 的“D:\VM_OS\ub100432bit\share”目录下看到这个文件。

4) 打开超级终端（超级终端配置参考“Tiny6410 设置超级终端.pdf”）。使用串口线把开发板的 COM0 口和 PC 的串口连接起来，把开发板的 S2 开关拨到 NAND 一侧，连接电源线，然后上电启动开发板，此时开发板上的 LINUX 系统开始启动，会在超级终端看到启动信息。

5) 等到开发板的 LINUX 系统启动完成后，使用超级终端，通过串口把 embed_ed 可执行文件下载到开发板：超级终端传送菜单-→发送文件，打开发送文件对话框，点击文件名后的浏览按钮，找到要发送的文件，协议选择 Zmodem 与崩溃恢复，点击发送，文件就发送到嵌入式 LINIUX 的当前目录下。注意：如果想重新发送某文件，应先把开发板上原来的文件删掉，然后再发送。新发送的文件不会覆盖开发板上原来的同名文件。

以下步骤所涉及到的操作，均针对开发板上的嵌入式 LINUX 操作系统，因此以下步骤涉及到的命令均在超级终端窗口下达。

6) 在超级终端窗口（此时的超级终端就是开发板上的 LINUX 的默认输出设备，可通过超级终端对开发板上的 LINUX 系统下达指令），输入如下命令：/etc/rc.d/init.d/leds stop。该命令将停止 led-player 对 led 的操纵（led-player 是嵌入式 LINUX 系统已经嵌入进去的 LED 点亮程序，该程序如不停止，那么用户开发的 LED 程序将不能调用 LED 设备）。

6) 在超级终端窗口，增加可执行权限 chmod+x embed_led，然后执行./embed_led。

7) 数据记录：观察记录 LED 灯的亮、灭情况，还可以在超级终端看到 LED 点亮情况的信息输出。

8) 修改程序，实现 LED 灯 1、4 亮，2、3 灭；延时后 2、3 亮，1、4 灭，循环往复。

9) 体会嵌入式 LINUX 下 LED 点亮程序和 ARM 裸机下 LED 灯点亮程序的区别，分析这两种不同程序设计之间的异同。

参考源程序

embed_led.c 源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
//延时函数
```

```
void delay()
{
	int i;
	for(i=0;i<70000000;i++);
}

int main()
{
	int fd;

	fd = open("/dev/leds", 0);//打开 LED 设备，返回文件描述符
	if (fd ==-1)
	{
		perror("open device leds");
		exit(1);
	}

	fprintf(stdout,"fd=%d\n",fd); //输出 LED 文件描述符

	while(1)
	{
		/*ioctl(fd, on, led_no)函数，用来控制 LED 灯。on=1，点亮 LED，on=0，熄灭 LED；
led_no 取值为 0、1、2、3，分别代表 LED1、LED2、LED3、LED4*/
		ioctl(fd,1,0);//LED1 点亮
		fprintf(stdout,"LED 1 ON\n");
		delay();
		ioctl(fd,0,0);//LED1 熄灭
		fprintf(stdout,"LED 1 OFF\n");
		delay();

		ioctl(fd,1,1); //LED2 点亮
		fprintf(stdout,"LED 2 ON\n");
		delay();
		ioctl(fd,0,1); //LED2 熄灭
		fprintf(stdout,"LED 2 OFF\n");
		delay();

		ioctl(fd,1,2); //LED3 点亮
		fprintf(stdout,"LED 3 ON\n");
		delay();
		ioctl(fd,0,2); //LED1 熄灭
		fprintf(stdout,"LED 3 OFF\n");
		delay();
```

```
            ioctl(fd,1,3); //LED4 点亮
            fprintf(stdout,"LED 4 ON\n");
            delay();
            ioctl(fd,0,3); //LED1 熄灭
            fprintf(stdout,"LED 4 OFF\n");
            delay();
        }
        close(fd);
        return 0;
}
```

Makefile：

```
# ---------------------------------------------------------------------------
# Makefile for building tapp
#
# Copyright 2010 FriendlyARM (http://www.arm9.net/)
#

ifndef DESTDIR
DESTDIR                 ?= /tmp/FriendlyARM/mini6410/rootfs
endif

CFLAGS                  = -Wall
CC                      = arm-linux-gcc
INSTALL                         = install

TARGET                  = embed_led


all: $(TARGET)

embed_led: embed_led.c
        $(CC) $(CFLAGS) $< -o $@


install: $(TARGET)
        $(INSTALL) $^ $(DESTDIR)/usr/bin

clean distclean:
        rm -rf *.o $(TARGET)


# ---------------------------------------------------------------------------
```

```
.PHONY: $(PHONY) install clean distclean

# End of file
# vim: syntax=make
```