实验 9 嵌入式 LINUX 下按键实验

实验目的

- 掌握嵌入式 LINUX 硬件设备编程方法，对按键进行编程使用。

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 linux_button_led.c，当按下 key1~key4 时，分别点亮 LED1~LED4，停顿一会后熄灭。
- 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

参考第 7 章相关部分

实验步骤

1) 在虚拟 UNBUNT 下编辑源程序 linux_button_led.c

2)编辑 Makefile 文件，执行 make，生成可执行文件 button_led。（或在虚拟 UNBUNT 下直接执行交叉编译 arm-linux-gcc，生成可执行文件 button_led）

3) 把 button_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 button_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 button_led 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）。

7)增加可执行权限 chmod+x button_led，然后执行./button_led。

8) 数据记录：分别按下开发板上的 key1~key4，观察记录 LED 的点亮情况。

9) 修改程序，实现如下功能：当 key1~key4 按下不放时，分别点亮 LED1~LED4，当 key1~key4 松开时，LED 熄灭。

10) 体会嵌入式 LINUX 下按键程序和 ARM 裸机下按键程序的区别，分析这两种不同程序设计之间的异同。

参考源程序

linux_button_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>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>

//延时函数
void delay()
{
    int i;
    for(i=0;i<70000000;i++);
}

int main(void)
{
    int fd_led,fd_button;
    char down='1';

    //打开 LED 设备
    fd_led = open("/dev/leds", 0);//打开 LED
    if (fd_led ==-1)
    {
        perror("open device leds");
        exit(1);
    }
```

```
//打开 key 设备
fd_button = open("/dev/buttons", 0);//打开 key
if (fd_button < 0)
{
	perror("open device buttons");
	exit(1);
}

while(1)
{
	char current_buttons[8];

	if (read(fd_button, current_buttons, sizeof current_buttons) != sizeof current_buttons)
	{
		perror("read buttons:");
		exit(1);
	}

	if(current_buttons[0]==down)//key1 按下
	{
		ioctl(fd_led,1,0);//LED1 点亮；
		delay();
		ioctl(fd_led,0,0);//LED1 熄灭；
	}

	if(current_buttons[1]==down)//key2 按下
	{
		ioctl(fd_led,1,1);//LED2 点亮；
		delay();
		ioctl(fd_led,0,1);//LED2 熄灭；
	}

	if(current_buttons[2]==down)//key3 按下
	{
		ioctl(fd_led,1,2);//LED3 点亮；
		delay();
		ioctl(fd_led,0,2);//LED3 熄灭；
	}

	if(current_buttons[3]==down)//key4 按下
	{
		ioctl(fd_led,1,3);//LED4 点亮；
		delay();
```

```
                ioctl(fd_led,0,3);//LED4 熄灭；
            }
        }

        close(fd_led);
        close(fd_button);

        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                  = button_led


all: $(TARGET)

button_led: linux_button_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
```