linux用户态使用gpio中断方法

一、用户空间 gpio 的调用文件

用户空间访问gpio，即通过sysfs接口访问gpio，下面是/sys/class/gpio目录下的三种文件：

–export/unexport文件

–gpioN指代具体的gpio引脚

–gpio_chipN指代gpio控制器

(1) export/unexport文件接口：

/sys/class/gpio/export，该接口只能写不能读

用户程序通过写入gpio的编号来向内核申请将某个gpio的控制权导出到用户空间当然前提是没有内核代码申请这个gpio端口

比如 echo 19 > export

上述操作会为19号gpio创建一个节点gpio19，此时/sys/class/gpio目录下边生成一个gpio19的目录

/sys/class/gpio/unexport和导出的效果相反。

比如 echo 19 > unexport

上述操作将会移除gpio19这个节点。

(2) /sys/class/gpio/gpioN

指代某个具体的gpio端口,里边有如下属性文件

direction：表示gpio端口的方向，读取结果是in或out。该文件也可以写，写入out 时该gpio设为输出同时电平默认为低。写入low或high则不仅可以设置为输出 还可以设置输出的电平。 当然如果内核不支持或者内核代码不愿意，将不会存在这个属性,比如内核调用了

gpio_export(N,0)就表示内核不愿意修改gpio端口方向属性

value：表示gpio引脚的电平,0(低电平)1（高电平）,如果gpio被配置为输出，这个值是可写的，记住任何非零的值都将输出高电平, 如果某个引脚能并且已经被配置为中断，则可以调用poll(2)函数监听该中断，中断触发后poll(2)函数就会返回。

edge：表示中断的触发方式，edge文件有如下四个值：“none”, “rising”, “falling”, “both”。

edge的值	含义
none	表示引脚为输入，不是中断引脚
rising	引脚为中断输入，上升沿触发
falling	引脚为中断输入，下降沿触发
both	引脚为中断输入，边沿触发

这个文件节点只有在引脚被配置为输入引脚的时候才存在。 当值是none时可以通过如下方法将变为中断引脚
echo “both” > edge;对于是both,falling还是rising依赖具体硬件的中断的触发方式。此方法即用户态gpio转换为中断引脚的方式

(3)/sys/class/gpio/gpiochipN
gpiochipN表示的就是一个gpio_chip,用来管理和控制一组gpio端口的控制器，该目录下存在一下属性文件：
base 和N相同，表示控制器管理的最小的端口编号。
lable 诊断使用的标志（并不总是唯一的）
ngpio 表示控制器管理的gpio端口数量（端口范围是：N ~ N+ngpio-1）

二、用户空间gpio操作方法
1.进入/sys/class/gpio目录

```
cd /sys/class/gpio
```

2.内核申请将某个gpio的控制权导出到用户空间，

```
echo 19 > export
```

3.看到出现gpio19的目录，进入该目录

```
cd gpio19
```

4.设置gpio控制方向输出

```
echo out > direction
```

5.设置gpio输出高电平

```
echo 1 > value
```

6.设置gpio输出低电平

```
echo 0 > value
```

7.设置gpio控制方向输入

```
echo in > direction
```

8.获取gpio的电平值

```
cat value
```

三、用户态使用gpio监听中断

1.设置gpio控制方向输入

```
echo in > direction
```

2.中断触发方式

```
echo  both > edge
```

3.使用poll函数监听中断，当然，上面的两个配置也可以直接写在函数中

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/poll.h>

int main(int argc, char *argv[])
{
    char buff[1024];
    int gpio_id;
    struct pollfd fds[1];
    int gpio_fd = open("/sys/class/gpio/gpio508/value", O_RDONLY);
    if (gpio_fd == -1)
        printf("gpio open");
    fds[0].fd = gpio_fd;
```

```
    fds[0].events = POLLPRI;
    int ret = read(gpio_fd, buff, 10);
    if (ret == -1)
        printf("read");

    while (1)
    {
        ret = poll(fds, 1, -1);
        if (ret == -1)
            printf("poll");

        if (fds[0].revents & POLLPRI)
        {
            ret = lseek(gpio_fd, 0, SEEK_SET);
            if (ret == -1)
                printf("lseek");
            ret = read(gpio_fd, buff, 10);
            if (ret == -1)
                printf("read");
            printf("get interrupt\n");
        }
    }
}
```