

# Linux如何进行GPIO读写操作的?



华为云开发者联盟

已认证帐号

3 人赞同了该文章

**摘要：**本文介绍GPIO的读写，介绍基本原理，以及不同读写方式的性能。

本文分享自华为云社区《[Linux 基于sysfs的GPIO读写操作](#)》，作者：一颗小树x。

## 前言

最近接触到Linux系统中的GPIO开发，这里做个小总结，也分享一下；本文会介绍GPIO的读写，介绍基本原理，以及不同读写方式的性能。

## 一、GPIO sysfs interface 基本原理

在Linux中，最常见的读写GPIO方式就是用GPIO sysfs interface，是通过操作/sys/class/gpio目录下的export、unexport、gpio{N}/direction, gpio{N}/value（用实际引脚号替代{N}）等文件实现的，经常出现shell脚本里面。

首先声明GPIO口，比如GPIO258：（这个命令需要root权限）

```
echo 258 > /sys/class/gpio/export
```

然后在/sys/class/gpio，会生成一个新目录，其名字叫gpio258

比如：GPIO258，N对应是258；进入其目录：cd /sys/class/gpio/gpio258

依次能看到：active\_low、device、direction、edge、power、subsystem、uevent、value

其中比较常用的是value文件，它存放GPIO的值；范围是：0或1。我们可以直接对它读写，达到读写GPIO的效果。

## 二、定义GPIO为输入

比如定义GPIO258为输入，命令如下：（这个命令需要root权限）

```
echo in > /sys/class/gpio/gpio258/direction
```

### 三、定义GPIO为输出

比如定义GPIO258为输出，命令如下：（这个命令需要root权限）

```
echo out > /sys/class/gpio/gpio258/direction
```

### 四、读GPIO的值

我们可以直接对/sys/class/gpio/gpio258 目录下的value文件（存放GPIO的值），进行读取。

可以用cat 查看读GPIO的值，比如查看GPIO258的值：

```
cat /sys/class/gpio/gpio258/value
```

value 只是一个文件，可以通过其他方式读取；写一个python程序，实现读取GPIO的值：

```
# 定义一个函数，用于读取GPIO258的值。
def read_258():
    with open('/sys/class/gpio/gpio258/value', 'r') as f:
        io_258 = int(f.read())
        print("read_258:%d"%io_258)

# 调用函数
read_258()
```

### 五、写GPIO的值

我们可以直接对/sys/class/gpio/gpio258 目录下的value文件（存放GPIO的值），进行写值。

可以用ehco 写GPIO的值，比如写GPIO258的值：

```
echo 1 > /sys/class/gpio/gpio258/value      # output logic 1 level
echo 0 > /sys/class/gpio/gpio258/value      # output logic 0 level
```

写一个python程序，实现写GPIO的值：

```
# 定义一个函数，用于写GPIO258的值。
def write_258(io_str):
    with open('/sys/class/gpio/gpio258/value', 'w+') as f:
        f.write(io_str)
        print("write_258:%s"%(io_str))

# 调用函数
write_258()
```

经过测试，程序进行一次写操作，耗时0.6ms左右；ehco方式就比较久了，10ms左右。

## 六、小案例——设置GPIO为输入，并读取IO值

### 方式一：纯shell命令

```
# 设置GPIO20为输入
echo 20 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio20/direction

# 读取IO值
cat /sys/class/gpio/gpio20/value
```

### 方式二：shell命令 + Python程序（效率更高）

```
# 设置GPIO20为输入
echo 20 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio20/direction
```

读取IO值：

```
# 定义一个函数，用于读取GPIO258的值。
def read_20():
    with open('/sys/class/gpio/gpio20/value', 'r') as f:
        io_20 = int(f.read())
        print("read_20:%d"%io_20)
```

```
# 调用函数
read_20()
```

## 七、小案例——设置GPIO为输出，并读写IO值

### 方式一：纯shell命令

```
# 设置GPIO40为输出
echo 40 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio40/direction

# 写IO值，高电平
echo 1 > /sys/class/gpio/gpio40/value

# 写IO值，低电平
echo 0 > /sys/class/gpio/gpio40/value
```

### 方式二：shell命令 + Python程序（效率更高）

```
# 设置GPIO40为输出
echo 40 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio40/direction
```

读写IO值：

```
import time

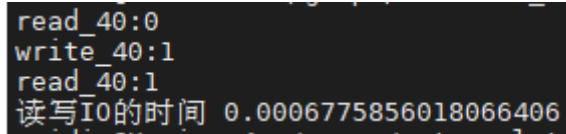
def read_40():
    with open('/sys/class/gpio/gpio40/value', 'r') as f:
        io_40 = int(f.read())
        print("read_40:%d"%io_40)

def write_40(io_str):
    with open('/sys/class/gpio/gpio40/value', 'w') as f:
        f.write(io_str)
        print("write_40:%s"%(io_str))

start = time.time()
read_40()
write_40("1")
```

```
read_40()  
end = time.time()  
print("读写IO的时间", end-start)
```

效果：耗时0.6ms。



```
read_40:0  
write_40:1  
read_40:1  
读写IO的时间 0.0006775856018066406
```