

Ubuntu例程

本文档介绍了如何在Ubuntu环境中读取Hi229/226 的数据，本路径提供了c语言例程代码，生成的可执行文件用于读取模块的数据。

测试环境： Ubuntu 16.04

测试设备：超核调试版 HI226 HI229

查找USB-UART设备

因为Ubuntu 系统自带CP210x的驱动，所以我们不用专门去安装相应串口驱动。将调试版连接到电脑上时，会自动识别设备。识别成功后，会在dev目录下出现一个对应的设备文件。

检查系统是否识别到USB-UART设备：

1、打开Ubuntu系统，按下 **ctrl + alt + t** 打开命令行窗口

2、在窗口上输入 `cd /dev` 切换到dev目录下，这个目录下，是一些设备文件。

3、然后在dev目录下执行 `ls` 这个命令是查看当前目录下都有哪些文件，然后按下 Enter 键，就会出现设备文件名称，在这些文件名称中，我们主要关心 **ttyUSB** 这个设备文件。后面数字代表USB设备号，由于Ubuntu USB设备号为从零开始依次累加，所以多个设备每次开机后设备号是不固定的，我们需要确定设备的设备号。下面我用两张图片来描述：

```
linux@ubuntu:~$ cd /dev
linux@ubuntu:/dev$ ls
agpgart      loop3      snapshot   tty33      tty7        ttyS8
autofs       loop4      snd         tty34      tty8        ttyS9
block        loop5      sr0         tty35      tty9        uhid
bsg          loop6      stderr      tty36      ttyprintk   uinput
btrfs-control loop7      stdin       tty37      ttyS0       urandom
bus          loop-control stdout       tty38      ttyS1       userio
cdrom        mapper     tty         tty39      ttyS10      vcs
cdwr         mcelog     tty0        tty4        ttyS11      vcs1
char         mem        tty1        tty40      ttyS12      vcs2
console      memory_bandwidth tty10       tty41      ttyS13      vcs3
core         midi        tty11       tty42      ttyS14      vcs4
cpu_dma_latency mqueue     tty12       tty43      ttyS15      vcs5
cuse         net         tty13       tty44      ttyS16      vcs6
disk         network_latency tty14       tty45      ttyS17      vcs7
dmideid      network_throughput tty15       tty46      ttyS18      vcsa
dri          null        tty16       tty47      ttyS19      vcsa1
dvd          port        tty17       tty48      ttyS2       vcsa2
ecryptfs     ppp         tty18       tty49      ttyS20      vcsa3
fb0          psaux       tty19       tty5        ttyS21      vcsa4
fd           ptmx        tty2         tty50      ttyS22      vcsa5
full         pts         tty20       tty51      ttyS23      vcsa6
fuse         random      tty21       tty52      ttyS24      vcsa7
hidraw0      rfkill      tty22       tty53      ttyS25      vfio
hpet         rtc         tty23       tty54      ttyS26      vga_arbiter
hugepages    rtc0        tty24       tty55      ttyS27      vhci
hwrng        sda         tty25       tty56      ttyS28      vhost-net
initctl      sda1        tty26       tty57      ttyS29      vhost-vsock
input        sda2        tty27       tty58      ttyS3       vmci
kmsg         sda5        tty28       tty59      ttyS30      vsock
lightnvme    sda6        tty29       tty6        ttyS31      zero
log          sda7        tty3         tty60      ttyS4
loop0        sg0         tty30       tty61      ttyS5
loop1        sg1         tty31       tty62      ttyS6
loop2        shm         tty32       tty63      ttyS7
linux@ubuntu:/dev$
```

上图为没有插入USB设备的情况，这个时候，dev目录下并没有名为 **ttyUSB** 文件，插入USB线，连接调试板，然后我们再次执行 `ls`：

dev目录下多了几个文件名称，如图：

```
linux@ubuntu:/dev$ ls
agpgart      loop3      shm        tty32       tty63       ttyS7
autofs       loop4      snapshot   tty33       tty7         ttyS8
block        loop5      snd        tty34       tty8         ttyS9
bsg          loop6      sr0        tty35       tty9         ttyUSB0
btrfs-control loop7      stderr     tty36       ttyprintk   ttyS10
bus          loop-control stdin      tty37       ttyS0       uinput
```

ttyUSB0 文件就是我们的调试板在ubuntu系统中生成的设备文件，对它进行读写，就可以完成串口通信。这个文件名称我们把它记下来。后面的数字是不固定的，有可能为 **ttyUSB1** 或 **ttyUSB2** 等。

编译并执行

我们开始在Ubuntu环境下生成一个可执行文件，专门用来解析模块的数据：

首先在Ubuntu系统中，按下 **ctrl + alt + t** 快捷键，在弹出的窗口上，执行 `mkdir hipnuc` 建立 hipnuc目录,然后将本文档所在目录下的所有文件复制到 **hipnuc** 目录下。

执行 `make`，生成可自行文件 `main`。并执行 `sudo ./main ttyUSB0`：

执行成功后，会出现这个画面：

```
device id: 0
frame rate: 50Hz
  Acc: 8      973      -222
  Gyo: -6      0        -2
  Mag: -67     -497     207
Eular(P R Y): -0.48  102.76  5.20
Please enter ctrl + 'c' to quit...
^C
```

这个画面上的数字会随着模块位置的改变而发生变化。

如果后期修改了这些文件，需要执行 `make clean` 命令，进行清理之前生成的旧 `.o` 和 `main` 文件，然后再执行 `make`，重新生成 `main` 这个可执行文件。

如果后期您需要在本路径上添加其他文件，配合使用，请打开 **Makefile** 文件，在第一行的后面加上后添加文件的链接文件名，例如添加 `append_file.c` 文件，那么在 **Makefile** 中第一行后面追加 `appen_file.o` 文件名。如果后加的文件还需要链接第三方的库，请在第二行的后面添加库名字。格式为 **-l+lib_name**。

如果出现：

```
open_port: Unable to open SerialPort: Bad file descriptor
Please enter usb port append to the execution command!!!
Please enter ctrl + 'c' to quit...
```

表示未能找到串口，需要回到《**查找USB-UART设备**》一节 确认USB-UART设备已经被ubuntu识别。