一 网口

理论支持

网络--双网卡--桥接方式->网桥

1. https://www.freebsd.org/doc/zh_CN.UTF-8/books/handbook/network-bridging.html

网桥的基本操作是将两个或多个网段连接在一起。 由于各式各样的原因， 人们会希望使用一台真正的计算机， 而不是网络设备来充任网桥的角色， 常见的原因包括线缆的限制、 需要进行防火墙， 或为虚拟机网络接口连接虚拟网络。 网桥也可以将无线网卡以 hostap 模式接入有线网络。

网桥可以用于连接两个不同的网段， 并用于监视往返的以太网帧。 这可以通过在网桥接口上使用 bpf(4)/tcpdump(1)， 或通过将全部以太网帧复制到另一个网络接口 (span 口) 来实现。

这一节主要介绍 if_bridge(4) 网桥实现。 除此之外， 还有一个基于 netgraph 的网桥实现， 如欲了解进一步细节， 请参见联机手册 ng_bridge(4)。

网桥驱动是一个内核模块， 并会随使用 ifconfig(8) 创建网桥接口时自动加载。 您也可以将 device if_bridge 加入到内核配置文件中， 以便将其静态联编进内核。

包过滤可以通过使用了 pfil(9) 框架的任意一种防火墙软件包来完成。 这些防火墙可以以模块形式加载， 也可以静态联编进内核。

通过配合 altq(4) 和 dummynet(4)， 网桥也可以用于流量控制。

32.5.4. 启用网桥

网桥是通过接口复制来创建的。 您可以使用 ifconfig(8) 来创建网桥接口， 如果内核不包括网桥驱动， 则它会自动将其载入。

```
# ifconfig bridge create

bridge0
```

```
# ifconfig bridge0

bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500

        ether 96:3d:4b:f1:79:7a

        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15

        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200

        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

如此就建立了一个网桥接口， 并为其随机分配了以太网地址。 maxaddr 和 timeout 参数能够控制网桥在转发表中保存多少个 MAC 地址， 以及表项中主机的过期时间。 其他参数控制生成树的运转方式。

将成员网络接口加入网桥。 为了让网桥能够为所有网桥成员接口转发包， 网桥接口和所有成员接口都需要处于启用状态:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up

# ifconfig fxp0 up

# ifconfig fxp1 up
```

网桥现在会在 fxp0 和 fxp1 之间转发以太网帧。 等效的 /etc/rc.conf 配置如下， 如此配置将在系统启动时创建同样的网桥。

```
cloned_interfaces="bridge0"

ifconfig_bridge0="addm fxp0 addm fxp1 up"

ifconfig_fxp0="up"

ifconfig_fxp1="up"
```

如果网桥主机需要 IP 地址， 则应将其绑在网桥设备本身， 而不是某个成员设备上。 这可以通过静态设置或 DHCP 来完成:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

除此之外， 也可以为网桥接口指定 IPv6 地址。

32.5.7. 网桥的高级用法

32.5.7.1. 重建流量流

网桥支持监视模式， 在 bpf(4) 处理之后会将包丢弃， 而不是继续处理或转发。 这可以用于将两个或多个接口上的输入转化为一个 bpf(4) 流。 在将两个独立的接口上的传输的 RX/TX 信号重整为一个时， 这会非常有用。

如果希望将四个网络接口上的输入转成一个流:

```
# ifconfig bridge0 addm fxp0 addm fxpl addm fxp2 addm fxp3 monitor up

# tcpdump -i bridge0
```

32.5.7.2. 镜像口 (Span port)

网桥收到的每个以太网帧都可以发到镜像口上。 网桥上的镜像口数量没有限制， 如果一个接口已经被配置为镜像口， 则它就不能再作为网桥的成员口来使用。 这种用法主要是为与网桥镜像口相连的监听机配合使用。

如果希望将所有帧发到名为 fxp4 的接口上:

```
# ifconfig bridge0 span fxp4
```

32.5.7.3. 专用接口 (Private interface)

专用接口不会转发流量到除专用接口之外的其他端口。 这些流量会无条件地阻断， 因此包括 ARP 在内的以太网帧均不会被转发。 如果需要选择性地阻断流量， 则应使用防火墙。

32.5.7.4. 自学习接口 (Sticky Interfaces)

如果网桥的成员接口标记为自学习， 则动态学习的地址项一旦进入转发快取缓存， 即被认为是静态项。 自学习项不会从快取缓存中过期或替换掉， 即使地址在另一接口上出现也是如此。 这使得不必事先发布转发表， 也能根据学习结果得到静态项的有点， 但在这些网段被网桥看到的客户机， 就不能漫游至另一网段了。

另一种用法是将网桥与 VLAN 功能连用， 这样客户网络会被隔离在一边， 而不会浪费 IP 地址空间。 考虑 CustomerA 在 v1an100 上， 而 CustomerB 则在 v1an101 上。 网桥地址为 192.168.0.1， 同时作为 internet 路由器使用。

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101

# ifconfig bridge0 inet 192.168.0.1/24
```

两台客户机均将 192.168.0.1 作为默认网关， 由于网桥快取缓存是自学习的， 因而它们无法伪造 MAC 地址来截取其他客户机的网络流量。

在 VLAN 之间的通讯可以通过专用接口（或防火墙）来阻断:

```
# ifconfig bridge0 private vlan100 private vlan101
```

这样这些客户机就完全相互隔离了。 可以使用整个的 /24 地址空间， 而无需划分子网。

2. http://blog.itist.tw/2014/05/hotspotd.html

橋接模式

適用於現有環境中已經有 IP 分享器、無線基地台或路由器在處理上網的服務，而我們只是希望使用 Raspberry Pi 來延伸現有的 Wi-Fi 訊號範圍或是提供原本不存在的 Wi-Fi 連線上網功能。

這個模式的重點是，透過 Raspberry Pi 的無線網卡連接上來的 Clients，會與現有環境中所有的 Clients 位於相同的 LAN 網段，也就是它們之間可以直接連線。

如果現有環境中沒有 DHCP Server 的話，請千萬不要使用橋接模式。

How to configure a Network Bridge on Ubuntu Server

http://www.havetheknowhow.com/Configure-the-server/Network-Bridge.html

3.SETTING UP A RASPBERRY PI AS A DHCP SERVER

http://www.noveldevices.co.uk/rp-dhcp-server

isc-dhcp-server

综上，以无线方式，实现 eth0 与 wlan0 的桥接 ，进而推进到有线网卡方式实现 eth0 与 eth1 的桥接。

http://dev.ardupilot.com/wiki/making-a-mavlink-wifi-bridge-using- HYPERLINK

德文:

Raspberry Pi als WLAN-Router einrichten (WLAN-Access-Point)

Mit Raspbian Jessie geprüft.

```
                    |------------------|
eth0[192.168.0.10]--|  raspi dhcp-srv  |--wlan0[192.168.0.10]
                    --------------------
```

Raspberry Pi als WLAN-Router einrichten (WLAN-Access-Point)

https://www.elektronik-kompendium.de/sites/raspberry-pi/2002171.htm

https://wiki.debian.org/BridgeNetworkConnections#Manual_bridge_setup

可实现无线 wifi

http://jacobsalmela.com/raspberry-pi-and-routing-turning-a-pi-into-a-router/

4. "2nd" serial and GPIO on raspi computer module

library BCM2835

http://www.airspayce.com/mikem/bcm2835/group__gpio.html

UART driver for Raspberry Pi

Raspberry Pi compute module has pinouts to use 2 UARTs : Mini and PL011. The full pledged PL011 UART has linux driver which apears as ttyAMA0.

The mini UART is a secondary low throughput UART intended to be used as a console. It isn't 16650 compatible.

Steps to cross-compile the driver on Ubuntu:

Download, build and install raspberry pi kernel 3.18.3+. (I could not find 3.12 kernel). Details here

Extract the contents of pi_mini_uart

Edit Makefile : change YOUR_CROSS_COMPILER_PREFIX (example : arm-none-linux-gnueabi-) and RASPBERRY_KERNEL_SOURCE_PATH (example: /home/user/pi_kernel)

run make

Change suitable GPIO for UART1 ALT function.

Mini UART will appear as ttyS0 after kernel module insertion.

The driver has following limitations:

No hardware flow control.

No console support.

Works only with 115200 8N1 configuration.

The documentation for BCM2835 has some glaring errors. Thanks to David Welch for the bare metal programs. Otherwise it would have been a very difficult task to get it working.

https://github.com/dwelch67/raspberrypi/blob/master/uart01/uart01.c

RS232 Serial on Raspberry Pi Compute Module

http://www.savagehomeautomation.com/projects/rs232-serial-on-raspberry-pi-compute-module.html#db9

http://www.savagehomeautomation.com/projects/ HYPERLINK "http://www.savagehomeautomation.com/projects/rs232-serial-on-raspberry-pi-compute-module.html" HYPERLINK "http://www.savagehomeautomation.com/projects/rs232-serial-on-raspberry-pi-compute-module.html"rs232-serial-on-raspberry-pi-compute-module.html

http://raspberrypi.stackexchange.com/questions/3475/how-to-get-more-than-one-uart-interface

"Software Serial"

Compute Module Attaching & Enabling Peripherals Guide

https://github.com/raspberrypi/documentation/blob/master/hardwa HYPERLINK

On a Compute Module both Bank0 and Bank1 are free to use with Bank2 used for eMMC and HDMI hot plug detect and ACT LED / USB boot control.

It then loads an ARM device tree file (e.g. bcm2708-rpi-cm.dtb for a Compute Module) and any device tree overlays specified in config.txt before starting the ARM subsystem and passing the device tree data to the booting Linux kernel.

raspi-update:

https://github.com/Hexxeh/rpi-update

84 MB Minimal Raspbian ARMHF Image for Raspberry Pi

109 MB Raspbian Wheezy armhf Raspberry Pi minimal image

222 MB Hexxeh image

http://raspberrypi.stackexchange.com/questions/5258/how-can-i-remove-the-gui-from-raspbian-debian

Devive tree for rapi computer module

https://github.com/raspberrypi/documentation/blob/master/configuration/device-tree.md

https://github.com/RPi-Distro/raspi-gpio/blob/master/raspi-gpio.c

具体实现

1.dhcp

sudo apt-get install isc-dhcp-server //dhcp的安装
//dpcp的配置
/etc/network/interfaces:
iface eth0 inet static
address <the-IP-address-of-your-Pi-that-will-be-the-DHCP-server>
netmask <the-subnet-mask-of-your-LAN>
gateway <the-IP-address-of-your-LAN-gateway>

/etc/dhcp/dhcp.conf:
subnet <starting-IP-address-of-your-network> netmask <starting-IP-address-of-your-network> {
range <first-IP-address-of-your-DHCP-address-range> <last-IP-address-of-your-DHCP-address-range>;
option routers <the-IP-address-of-your-gateway-or-router>;
option broadcast-address <the-broadcast-IP-address-for-your-network>;
}

/etc/default/isc-dhcp-server:

DHCPD_CONF=/etc/dhcp/dhcpd.conf
DHCPD_PID=/var/run/dhcpd.pid
INTERFACES="eth0"

sudo service isc-dhcp-server restart //dpcp的重启

2.无线桥接

二串口

GPIO_List
https://github.com/rewolff/bw_rpi_tools/blob/master/gpio/gpio_list.c

Serial
http://blackeagle12.net/Comp/RPi/Serial.html

http://www.instructables.com/id/Read-and-write HYPERLINK "http://www.instructables.com/id/Read-and-write-from-serial-port-with-Raspberry-Pi/"-from-serial-port-with-Raspberry-Pi/

You have to disable the serial getty service:
CODE: SELECT ALL
sudo systemctl stop serial-getty@ttyAMA0.service
https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=123081

you can also disable it from start again:

CODE: SELECT ALL
sudo systemctl disable serial-getty@ttyAMA0.service

使能串口的login功能
http://www.hobbytronics.co.uk/raspberry-pi-serial-port
first and main one is /etc/inittab
Disable it by adding a # character to the beginning. Save the file.

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
second file /boot/cmdline.txt
The contents of the file look like this
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
去掉红色的部分
Ser2net
Ser2net Configuration
You must configure the ser2net as shown below.

Open the config file for edit:
sudo gedit /etc/ser2net.conf

Comment out the unneeded entries near the end of the file with # character.

Add a line like this:

192.168.1.104,2000:raw:0:/dev/ttyAMA0:9600 8DATABITS NONE 1STOPBIT banner

MavProxy

http://diydrones.com/profiles/blogs/step-by-step-guide-to-mavproxy-on-windows-7-live-forward-your-uav?xg_source=activi HYPERLINK
MAVProxy is powerful ground station software that excellently complements your favorite GUI ground station, such as Mission Planner, APM Planner etc. A key driver for me to look into MAVProxy was it's capability to forward the signal from your UAV over the network via UDP to multiple other ground station software on other devices. For example; you can run a ground station on a laptop next to your antenna and forward via wifi to a smartphone/tablet which lets you easily relocate to launch into wind before heading back to your fixed antenna. I have also used it to send telemetry data to a friend acting as spotter several kilometers away (via 4G vpn) during a longer flight so that he could monitor the entire flight and determine where to look to find the aircraft in flight.

Alas it is not the easiest to understand how to run on Windows and I had no luck finding a single step by step guide for the non technical, so I have had a go at creating one. This guide will let you successfully set up MAVProxy to allow forwarding via network interfaces and usage via command line. There may be other ways to get this running and you may need other packages as per the official MAVProxy documentation at
http://tridge.githu HYPERLINK "http://tridge.github.io/MAVProxy/" HYPERLINK

in order to use more advanced functions. No warranty responsibility for damage etc. Full credit to Andrew Tridgell and all other contributors to MAVProxy and the other software used here.

Step 7 - Forwarding over network
To forward the MAV data over the network including to a local program on your PC we simply need to add some extra parameters when starting MAVProxy via the command line.
To connect with a local ground station software such as Mission Planner start MAVProxy as above with the command mavproxy.py --master="com14" --baudrate 57600 --out 127.0.0.1:14550 and press enter.
Then open Mission Planner and select UDP and click connect. Click OK on the default prompt for port number (14550) and you should see mission planner start downloading parameters and connecting to your UAV.
Finally you can add the IP address of any computer to forward the telemetry stream onwards to other ground stations.
1) On the local network/wifi you will need to ensure there is no firewall on the client PC stopping the incoming stream to your ground station software.
2) Add --out IP_ADDRESS:14550 to the end of the mavproxy.py command. You can add as many separate --out parameters as you want depending on how many extra ground stations you are running.
3) Set each ground station to listen for UDP packets on port 14550

python

工程

serial - wiringSerial http://wiringpi.com/reference/serial-library/

```
#include <errno.h>

#include <wiringPi.h>
#include <wiringSerial.h>

int main ()
{
 int fd ;
 int count ;
 unsigned int nextTime ;

 if ((fd = serialOpen ("/dev/ttyAMA0",57600 )) < 0)
 {
   fprintf (stderr, "Unable to open serial device: %s\n", strerror
```

```
(errno)) ;
      return 1 ;
    }

    if (wiringPiSetup () == -1)
    {
      fprintf (stdout, "Unable to start wiringPi: %s\n", strerror (errno)) ;
      return 1 ;
    }

    nextTime = millis () + 300 ;

    for (count = 0 ; count < 256 ; )
    {
      if (millis () > nextTime)
      {
  //      printf ("\nOut: %3d: ", count) ;
  //      fflush (stdout) ;
  //      serialPutchar (fd, count) ;
  //      nextTime += 300 ;
  //      ++count ;
      }

      delay (13) ;

      while (serialDataAvail (fd))
      {
        printf (" %3x ", serialGetchar (fd)) ;
        fflush (stdout) ;
      }
    }

    printf ("\n") ;
    return 0 ;
  }
```

//==========安装步骤:

1. 热点的安装配置，参考

http://jacobsalmela.com/raspberry-pi-and-routing-turning-a-pi-into-a-router/

https://github.com/le HYPERLINK "https://github.com/lee7/raspberry-pi-router" HYPERLINK "https://github.com/lee7/raspberry-pi-router" HYPERLINK "https://github.com/lee7/raspberry-pi-router" HYPERLINK "https://github.com/lee7/raspberry-pi-router"e7/raspberry-pi-router

```
sudo vi /etc/apt/sources.list
```

添加:

deb http://mirror.sysu.edu.cn/raspbian/raspbian/

deb http://mirrors.neusoft.edu.cn/raspbian/raspbian/

```
sudo apt-get update
sudo apt-get upgrade
```

Install Required Components

rfkill: a wireless utility

zd1211-firmware: common firmware that works with many Wi-Fi dongles

hostapd: the hostap wireless access point daemon

hostap-utils: supplemental hostap tools

iw: wireless configuration utility

dnsmasq: DHCP and DNS utility

bridge-utils: used for connecting multiple Ethernet devices together

```
sudo apt-get install rfkill zd1211-firmware hostapd hostap-utils iw dnsmasq
bridge-utils
sudo cp /etc/network/interfaces /etc/network/interfaces.orig
sudo vi /etc/network/interfaces
```

Modify the file as below (highlighted lines are the additions):

```
auto lo
auto br0

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
allow-hotplug eth0
iface wlan0 inet manual

iface br0 inet dhcp
        bridge_fd 1
        bridge_hello 3
        bridge_maxage 10
```

```
        bridge_stp off
        bridge_ports eth0 wlan0
```

```
sudo ifdown wlan0
sudo ifup wlan0
sudo cp /etc/hostapd/hostapd.conf /etc/hostapd/hostapd.conf.orig
sudo vi /etc/hostapd/hostapd.conf
```

modify the file as follows:

```
interface=wlan0
bridge=br0
driver=rtl871xdrv
country_code=US
ctrl_interface=wlan0
ctrl_interface_group=0
ssid=RPiAP
hw_mode=g
channel=1
wpa=3
wpa_passphrase=password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
beacon_int=100
auth_algs=3
macaddr_acl=0
wmm_enabled=1
eap_reauth_period=360000000
```

```
sudo cp /etc/default/hostapd /etc/default/hostapd.orig
sudo vi /etc/default/hostapd
```

and then modify the highlighted line below (be sure to uncomment the line):

```
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd
configuration
```

```
# file and hostapd will be started during system boot. An example
configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"

# Additional daemon options to be appended to hostapd command:-
#       -d   show more debug messages (-dd for even more)
#       -K   include key data in debug messages
#       -t   include timestamps in some debug messages


sudo cp /usr/sbin/hostapd /usr/sbin/hostapd.orig
cd /usr/sbin
sudo rm -f hostapd

sudo wget http://www.daveconroy.com/wp3/wp-content/uploads/2013/07/hostapd.zip
[http://www.daveconroy.com/turn-your-raspberry-pi-into-a-wifi-hotspot-with-
edimax-nano-usb-ew-7811un-rt18188cus-chipset/
]

sudo chown root:root hostapd
sudo chmod 755 hostapd
sudo service networking restart
sudo service hostapd restart

sudo cp /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo vi /etc/dnsmasq.conf

Uncomment the following lines and adjust them to your environment:
domain-needed
interface=wlan0
dhcp-range=192.168.2.1,192.168.2.254,12h

pi
```

2.wiringPi 库的安装

http://wiringpi.com/reference/serial-library/

http://wiringpi.com/download-and-install/

https://github.com/WiringPi/WiringPi

git clone https://github.com/WiringPi/WiringPi

cd wiringPi

./build

编译要加 -lwiringPi

```
#include <stdio.h>
#include <string.h>
#include <errno.h>

#include <wiringPi.h>
#include <wiringSerial.h>

int main ()
{
  int fd,fd0;

  if ((fd = serialOpen ("/dev/ttyAMA0",57600 )) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;
  }

if((fd0 = serialOpen("/dev/ttyUSB0",57600))<0)
{
   fprintf(stderr,"Unable to open usb device: %s\n",strerror (errno));
   return 1;
}

  if (wiringPiSetup () == -1)
  {
    fprintf (stdout, "Unable to start wiringPi: %s\n", strerror (errno)) ;
    return 1 ;
  }

 // for (count = 0 ; count < 256 ; )
 while(1)
  {

    // delay (13) ;

    if(serialDataAvail (fd))
    {
    //   printf (" %x ", serialGetchar (fd)) ;
    //  fflush (stdout) ;

    //  printf("send to usb0\n");
```

```
      serialPutchar (fd0, serialGetchar (fd)) ;

    }

      if(serialDataAvail (fd0))
    {
      // printf (" %x ", serialGetchar (fd0)) ;
     // fflush (stdout) ;

      // printf("send to AMA0\n");
      serialPutchar (fd, serialGetchar (fd0)) ;

    }
  }

  // printf ("\n") ;
  return 0 ;
}
```

//会有闪退问题，mavproxy 是如何实现不闪退的？？？？

3. MAVProxy+?ser2net （不需要 ser2net）（查找 MAVProxy 源码）

http://dronecode.github.io/MAVProxy/html/getting_started/quickstart.html

http://dronecode.github.io/MAVProxy/html/getting_started/download_and_installation.html#linux

http://dev.ardupilot.com/wiki/making-a-mavlink-wifi-bridge-using-the-raspberry-pi/#testing_the_mavproxy_connection

```
sudo apt-get install python-opencv python-wxgtk2.8 python-pip python-dev

sudo pip install MAVProxy
```

Configuring MavProxy to always run and listen to incoming connections

The next step to get this working is to setup MavProxy to run automatically with the RPi boots up. To do this MavProxy has a daemon mode that works similarly to the above configuration for the DHCP server. In order to set it up we will use a script and modify it to work with the RPi.

Download the mavgateway file (from Github) and then copy it over to to the RPi.

wget https://raw.githubusercontent.com/dronekit/dronekit-python/3222b2cdb71b8b4c5b924ee28fc5d3529be00e67/scripts/mavgateway

We now have to edit the file and change some aspects of it to make it compatible. Lets edit the file:

To setup the correct parameters to start MavProxy find the following line:

```
DAEMON_ARGS="--master=/dev/ttyMFD1,115200 --out=udpin:0.0.0.0:14550 --daemon"
```

If you're using a serial connection change it to this:

```
DAEMON_ARGS="--master=/dev/ttyAMA0,57600 --out=udpin:0.0.0.0:14550 --daemon"
```

If you're using a USB connection change it to this:

```
DAEMON_ARGS="--master=/dev/ttyUSB0,57600 --out=udpin:0.0.0.0:14550 --daemon"
```

改为

```
DAEMON_ARGS="--master=/dev/ttyAMA0,57600 --out=udpin:0.0.0.0:14550
--out=/dev/ttyUSB0,57600 --daemon"
```

Next we need to change the user that starts MavProxy, find this line:

```
start-stop-daemon --start --background --make-pidfile --chuid edison
--chdir /tmp --quiet --pidfile $PIDFILE --exec $DAEMON -- \
```

And change it to:

```
start-stop-daemon --start --background --make-pidfile --chuid pi --chdir /tmp
--quiet --pidfile $PIDFILE --exec $DAEMON -- \
```

Now we need to move the file and set the correct permissions. Type the following commands:

```
sudo mv mavgateway /etc/init.d/mavgateway
```

```
sudo chown root:root mavgateway
```

```
sudo chmod 755 mavgateway
```

We are now ready to setup the daemon to run every time the RPi boots type this command:

```
sudo update-rc.d mavgateway defaults
```

Reboot the RPi:

```
sudo reboot
```

Connecting to the MavGateway MavProxy（不需要）

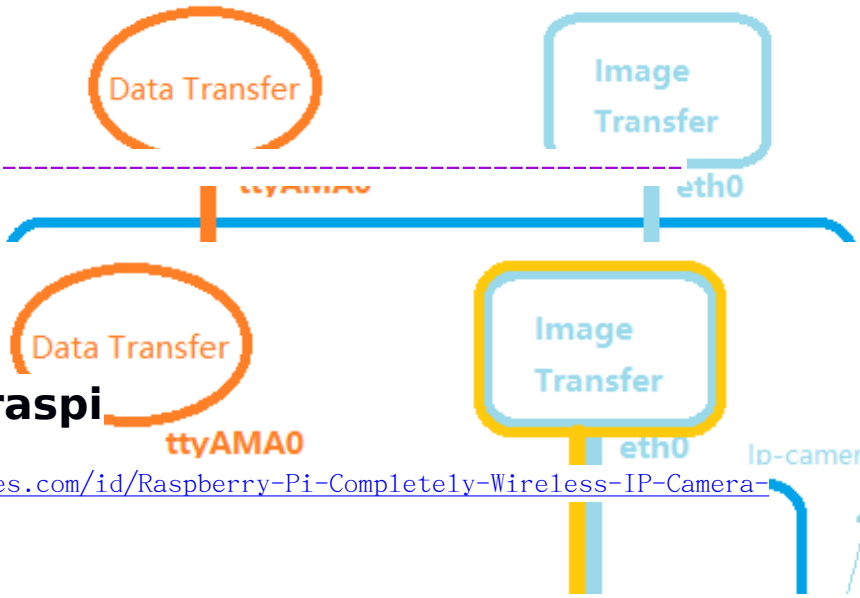
Using MavProxy (replace xxx.xxx.xxx.xxx with the IP Address of the RPi):

```
mavproxy.py —master=udpout:192.168.137.100:14550
```

问题：usb 连接方式会闪退，笔记本如果不接收，会重启，原因未知。

换了个 usb 转串口 pc 重启的问题好像不存在了？！

A----------



B----------

4. Ip camera raspi

http://www.instructables.com/id/Raspberry-Pi-Completely-Wireless-IP-Camera-Solar/

配置 raspi-A 网络桥接 eth0-br-eth1

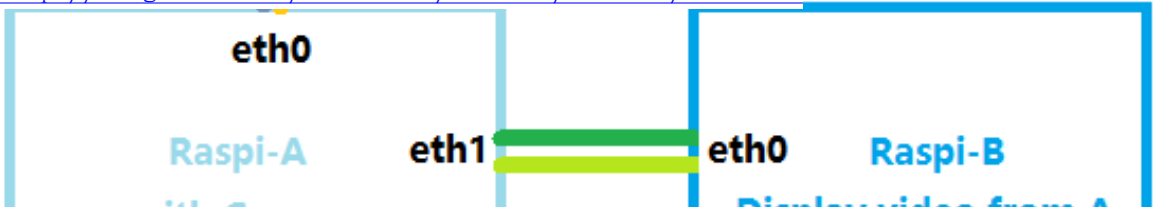
DMI

raspi # gstreamer - tcpclientsink 和 udpsrc 插件用法

http://www.cnblogs.com/kkia/p/3816444.html

VLC 获取 Raspberry Pi 使用 UV4L HTTP Streaming Server 提供的实时视频流

http://blog.csdn.net/hn11c2012/article/details/46347349



gstreamer

```
sudo apt-get install gstreamer1.0
```

http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-camera/streaming-video-using-gstreamer

```
 raspivid --verbose --nopreview -hf -vf --width 640 --height 480 --framerate 15 --bitrate 1000000 --profile baseline --timeout 0 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=1 pt=96 ! udpsink host=192.168.137.255 port=5700
```

```
[device(tower) ip]
```

255广播地址

https://github.com/DroidPlanner/Tower/wiki/Custom- HYPERLINK "https://github.com/DroidPlanner/Tower/wiki/Custom-video-stream"video-stream

1> flash emmc

https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md

Git may produce an error if the date is not set correctly, so on a Raspberry Pi enter the following:

```
sudo date MMDDhhmm
```

where MM is month, DD day and hh mm hours and minutes respectively.

Clone the usbboot tool repository:

```
git clone --depth=1 https://github.com/raspberrypi/tools
cd tools/usbboot
```

libusb must be installed. If you are using Cygwin, please make sure libusb is installed as previously described. On the Raspberry Pi or other Debian-based Linux enter the following command:

```
sudo apt-get install libusb-1.0-0-dev
```

Now build and install the usbboot tool:

```
make
sudo make install
```

Run the usbboot tool and it will wait for a connection:

```
sudo rpiboot
```

Now plug the host machine into the Compute Module IO board USB slave port (J15) and power on the CMIO board. The rpiboot tool will discover the Compute Module and send boot code to allow access to the eMMC.

2> spi->net

http://raspi.tv/2015/ethernet-on-pi-zero-how-to-put-an-ethernet-port-on-your-pi

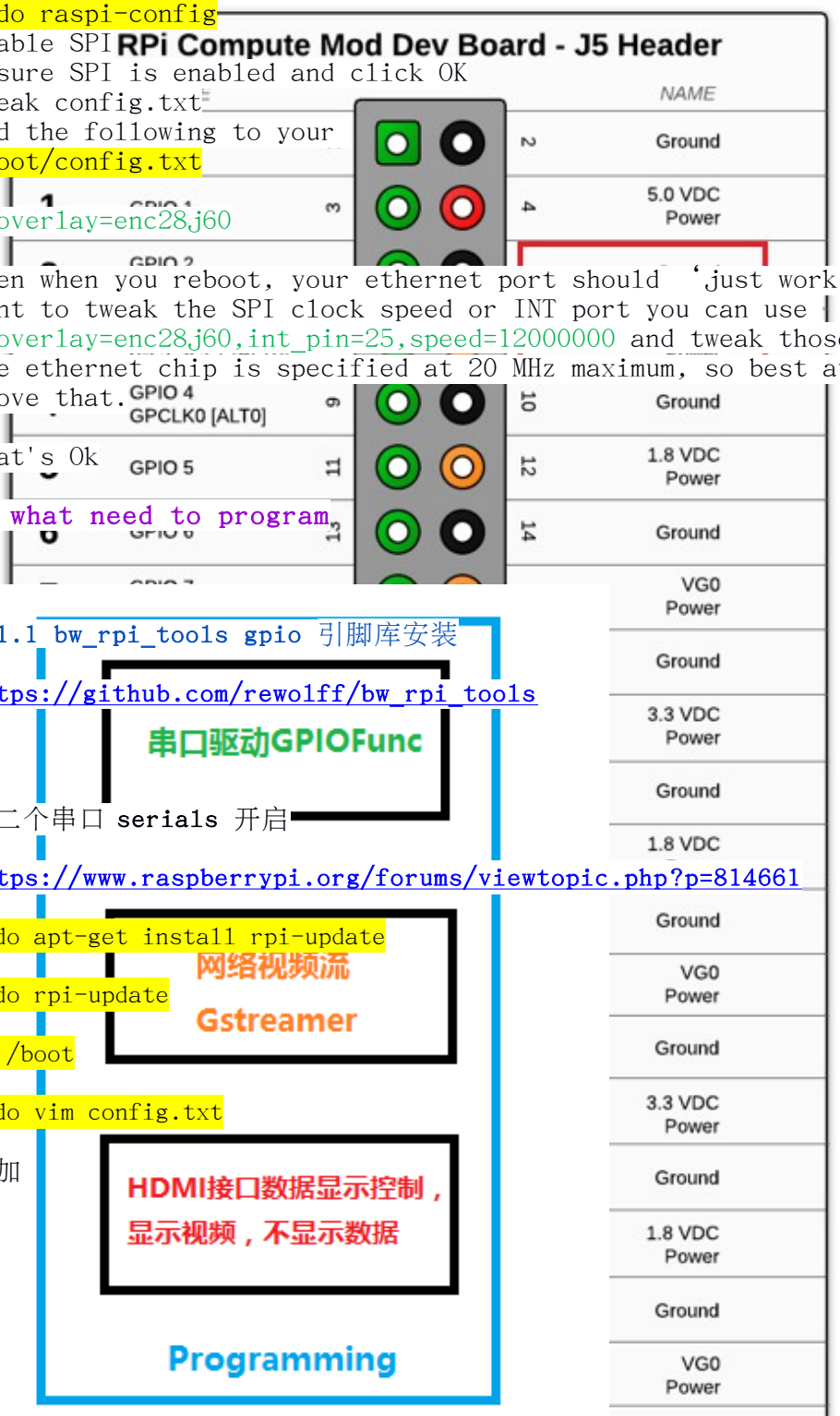
sudo raspi-config
enable SPI
Ensure SPI is enabled and click OK
Tweak config.txt
Add the following to your /boot/config.txt

dtoverlay=enc28j60

Then when you reboot, your ethernet port should 'just work'. If you want to tweak the SPI clock speed or INT port you can use dtoverlay=enc28j60,int_pin=25,speed=12000000 and tweak those variables. The ethernet chip is specified at 20 MHz maximum, so best avoid going above that.

That's Ok

3> what need to program



1.1.1 bw_rpi_tools gpio 引脚库安装

https://github.com/rewolff/bw_rpi_tools

串口驱动GPIOFunc

第二个串口 serials 开启

https://www.raspberrypi.org/forums/viewtopic.php?p=814661

sudo apt-get install rpi-update

网络视频流

sudo rpi-update

Gstreamer

cd /boot

sudo vim config.txt

添加

HDMI接口数据显示控制，显示视频，不显示数据

Programming

```
# At the bottom I added this
dtoverlay=uart1,txd1_pin=40,rxd1_pin=41
sudo reboot
```

问题如何同时启动 spi2net？？

```
sudo apt-get install device-tree-compiler
in /boot
vim uart1enc28j60-overlay.dts
uart1enc28j60-overlay.dts
//================code begin
// Overlay for the Microchip ENC28J60 Ethernet Controller
/dts-v1/;
/plugin/;

/ {
    compatible = "brcm,bcm2708";

    fragment@0 {
        target = <&spi0>;
        __overlay__ {
            /* needed to avoid dtc warning */
            #address-cells = <1>;
            #size-cells = <0>;

            status = "okay";

            spidev@0{
```

```
            status = "disabled";
        };

        eth1: enc28j60@0{
            compatible = "microchip,enc28j60";
            reg = <0>; /* CE0 */
            pinctrl-names = "default";
            pinctrl-0 = <&eth1_pins>;
            interrupt-parent = <&gpio>;
            interrupts = <25 0x2>; /* falling edge */
            spi-max-frequency = <12000000>;
            status = "okay";
        };
    };
};

fragment@1 {
    target = <&gpio>;
    __overlay__ {
        eth1_pins: eth1_pins {
            brcm,pins = <25>;
            brcm,function = <0>; /* in */
            brcm,pull = <0>; /* none */
        };
    };
};
```

```
fragment@2 {
    target = <&uart1>;
    __overlay__ {
        pinctrl-names = "default";
        pinctrl-0 = <&uart1_pins>;
        status = "okay";
    };
};

fragment@3 {
    target = <&gpio>;
    __overlay__ {
        uart1_pins: uart1_pins {
            brcm,pins = <14 15>;
            brcm,function = <2>; /* alt5 */
            brcm,pull = <0 2>;
        };
    };
};

fragment@4 {
    target-path = "/chosen";
    __overlay__ {
        bootargs = "8250.nr_uarts=1";
    };
};
```

```
    __overrides__ {
        int_pin = <&eth1>, "interrupts:0",
                  <&eth1_pins>, "brcm,pins:0";
        speed   = <&eth1>, "spi-max-frequency:0";
        txd1_pin = <&uart1_pins>,"brcm,pins:0";
        rxd1_pin = <&uart1_pins>,"brcm,pins:4";
    };
};
//============code end
```

sudo dtc -@ -I dts -O dtb -o /boot/overlays/uart1enc28j60-overlay.dtb /boot/uart1enc28j60.dts

Edit /boot/config.txt and add the line:

dtoverlay=uart1enc28j60,txd1_pin=40,rxd1_pin=41

6. USB 传输 UDP 格式的视频流

http://elinux.org/How_to_use_an_Android_tablet_as_a_Raspberry_Pi_console_terminal_and_internet_router

Enable Android USB tethering

Enable USB tethering on your Android tablet.

Settings → Connections → Tethering and Wi-Fi hotspot → USB tethering

Now the Raspberry automatically gets an IP address via its USB0 port. The Android tablet behaves like a NAT router. This should work both when your tablet is connected to Wi-Fi or to a mobile 3G/4G network.

Network setup

sudo vim /etc/network/interfaces

add

```
iface usb0 inet static
```

```
address 192.168.42.42
netmask 255.255.255.0
network 192.168.42.0
broadcast 192.168.42.255
```

```
sudo dhclient usb0
```

on the PI should get the ip from phone and use it as router

```
ifconfig
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4010 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4010 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:168456 (164.5 KiB)  TX bytes:168456 (164.5 KiB)
usb0      Link encap:Ethernet  HWaddr 02:07:00:01:62:31
          inet addr:192.168.42.159  Bcast:192.168.42.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19830 errors:0 dropped:1 overruns:0 frame:0
          TX packets:10412 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:21499431 (20.5 MiB)  TX bytes:2806515 (2.6 MiB)
```

Tower 增加 usb tetheting 设置功能
在 flightActivity 的 onCreate() 中添加如下代码:

```
Intent tetherSettings = new Intent();
tetherSettings.setClassName("com.android.settings",
"com.android.settings.TetherSettings");
startActivity(tetherSettings);
//=====================end by tgl
```

http://stackoverflow.com/questions/9913645/android-enable-usb-tethering-programmatically-there-is-an-app-that-did-it-fo
http://stackoverflow.com/questions/14340084/enable-usb-tethering-android-programmatically-without-user-interaction

获取 ipcamera 视频流

```
gst-launch-1.0 rtspsrc location=rtsp://192.168.0.123:554/mpeg4 latency=250 !
rtph264depay ! avdec_h264 ! autovideosink fps-update-interval=1000 sync=false
```

raspi 下 usb 摄像头的显示

http://www.sibilabs.com/2012/09/26/rpi-webcam-stream/

方式一 通过 luvcview 工具 `sudo apt-get install luvcview`
在 destop 下命令行里输入:

```
luvcview -d /dev/video0 -f yuv -s 640x480
```

方式二 fswebcam https://www.raspberrypi.org/documentation/usage/webcams/

HDMI 支持热插热拔功能

http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-

HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK "http://raspberrypi.stackexchange.com/questions/2169/how-do-i-force-the-raspberry-pi-to-turn-on-hdmi" HYPERLINK

Add these two lines to /boot/config.txt and reboot Raspbmc:

```
hdmi_force_hotplug=1
hdmi_drive=2
```

hdmi_force_hotplug=1 sets the Raspbmc to use HDMI mode even if no HDMI monitor is detected. hdmi_drive=2 sets the Raspbmc to normal HDMI mode (Sound will be sent if supported and enabled). Without this line, the Raspbmc would switch to DVI (with no audio) mode by default.

HDMI 输出视频流:

```
raspivid -g 10 -o - -w 800 -h 600 -ih -vf -hf -t 0 -n -fps 24 |
 gst-launch-1.0 fdsrc fd=0 ! \
        'video/x-h264,framerate=25/1' ! \
        h264parse ! \
        mpegtsmux ! \
        filesink location=/dev/stdout | \
        omxplayer -b --no-keys --live pipe:0
```

https://www.raspberrypi.org/forums/viewtopic.php?t=102887

Linux C 中如何调用 shell:

http://my.oschina.net/renhc/blog/53580

1) system(shell 命令或 shell 脚本路径):

system()会调用 fork()产生 子历程, 由子历程来调用/bin/sh-c string 来履行 参数 string 字符串所代表的命令, 此命令履行 完后随即返回原调用的历程。在调用 system()期间 SIGCHLD 信号会被暂时搁置, SIGINT 和 SIGQUIT 信号则会被漠视 。

返回值: 如果 system()在调用/bin/sh 时失败则返回 127, 其他失败原因返回-1。若参数 string 为空指针(NULL), 则返回非零值。 如果 system()调用成功 则最后会返回履行 shell 命令后的返回值, 但是此返回值也有可能为 system()调用/bin/sh 失败所返回的 127, 因 此最好能再反省 errno 来确认履行 成功 。

system 命令以其简略 高效的作用得到很很广泛 的利用

2) popen()会调用 fork()产生 子历程, 然后从子历程中调用/bin/sh -c 来履行 参数 command 的指令。参数 type 可应用 “r” 代表读取, “w” 代表写入。遵循此 type 值, popen()会建立 管道连到子历程的标准 输出设备 或标准 输入设备 , 然后返回一个文件指针。随后历程便可利用 此文件指针来读取子历程的输出设备 或是写入到子历程的标准 输入设备 中。此外, 所有应用 文 件指针(FILE*)操作的函数也都可以应用 , 除了 fclose()以外。

返回值: 若成功 则返回文件指针, 否则返回 NULL, 差错 原因存于 errno 中。注意: 在编写具 SUID/SGID 权限的程序时请尽量避免应用 popen(), popen()会继承环境变量, 通过环境变量可能会

造成系统安全的问题。
http://my.oschina.net/renhc/blog/35116
为了更好的理解 system()函数返回值，需要了解其执行过程，实际上 system()函数执行了三步操作:
1.fork 一个子进程;
2.在子进程中调用 exec 函数去执行 command;
3.在父进程中调用 wait 去等待子进程结束。
对于 fork 失败，system()函数返回-1。
如果 exec 执行成功，也即 command 顺利执行完毕，则返回 command 通过 exit 或 return 返回的值。（注意，command 顺利执行不代表执行成功，比如 command: "rm debuglog.txt"，不管文件存不存在该 command 都顺利执行了）
如果 exec 执行失败，也即 command 没有顺利执行，比如被信号中断，或者 command 命令根本不存在，system()函数返回 127.
如果 command 为 NULL，则 system()函数返回非 0 值，一般为 1.

曾经的曾经，被 system()函数折磨过，之所以这样，是因为对 system()函数了解不够深入。只是简单的知道用这个函数执行一个系统命令，这远远不够，它的返回值、它所执行命令的返回值以及命令执行失败原因如何定位，这才是重点。当初因为这个函数风险较多，故抛弃不用，改用其他的方法。这里先不说我用了什么方法，这里必须要搞懂 system()函数，因为还是有很多人用了 system()函数，有时你不得不面对它。

先来看一下 system()函数的简单介绍:
?
```
#include <stdlib.h>
int system(const char *command);
system() executes a command specified in command by calling /bin/sh -c command, and returns after the command has been completed. During execution of the command, SIGCHLD will be blocked, and SIGINT and SIGQUIT will be ignored.
```

system()函数调用/bin/sh 来执行参数指定的命令，/bin/sh 一般是一个软连接，指向某个具体的 shell，比如 bash，-c 选项是告诉 shell 从字符串 command 中读取命令;
在该 command 执行期间，SIGCHLD 是被阻塞的，好比在说：hi，内核，这会不要给我送 SIGCHLD 信号，等我忙完再说;
在该 command 执行期间，SIGINT 和 SIGQUIT 是被忽略的，意思是进程收到这两个信号后没有任何动作。

再来看一下 system()函数返回值:
```
The value returned is -1 on error (e.g. fork(2) failed), and the return status of the command otherwise. This latter return status is in the format specified in wait(2). Thus, the exit code of the command will be WEXITSTATUS(status). In case /bin/sh could not be executed, the exit status will be that of a command that does exit(127).
If the value of command is NULL, system() returns nonzero if the shell is available, and zero if not.
```
为了更好的理解 system()函数返回值，需要了解其执行过程，实际上 system()函数执行了三步操作:
1.fork 一个子进程;
2.在子进程中调用 exec 函数去执行 command;
3.在父进程中调用 wait 去等待子进程结束。
对于 fork 失败，system()函数返回-1。
如果 exec 执行成功，也即 command 顺利执行完毕，则返回 command 通过 exit 或 return 返回的值。（注意，command 顺利执行不代表执行成功，比如 command: "rm debuglog.txt"，不管文件存不存在该 command 都顺利执行了）
如果 exec 执行失败，也即 command 没有顺利执行，比如被信号中断，或者 command 命令根本不存在，

system()函数返回127.
如果command为NULL，则system()函数返回非0值，一般为1.

看一下system()函数的源码
看完这些，我想肯定有人对system()函数返回值还是不清楚，看源码最清楚，下面给出一个system()函数的实现:

```
33
int system(const char * cmdstring)
{
    pid_t pid;
    int status;

if(cmdstring == NULL)
{
    return (1); //如果cmdstring为空，返回非零值，一般为1
}

if((pid = fork())<0)
{
    status = -1; //fork失败，返回-1
}
else if(pid == 0)
{
    execl("/bin/sh", "sh", "-c", cmdstring, (char *)0);
    _exit(127); // exec执行失败返回127，注意exec只在失败时才返回现在的进程，成功的话现在的进程就不存在啦~~
}
else //父进程
{
    while(waitpid(pid, &status, 0) < 0)
    {
        if(errno != EINTR)
        {
            status = -1; //如果waitpid被信号中断，则返回-1
            break;
        }
    }
}

    return status; //如果waitpid成功，则返回子进程的返回状态
}
```

仔细看完这个system()函数的简单实现，那么该函数的返回值就清晰了吧，那么什么时候system()函数返回0呢？只在command命令返回0时。

看一下该怎么监控system()函数执行状态
这里给我出的做法:

```
23
24
int status;
if(NULL == cmdstring) //如果cmdstring为空趁早闪退吧，尽管system()函数也能处理空指针
{
    return XXX;
}
status = system(cmdstring);
```

```
if(status < 0)
{
    printf("cmd: %s\t error: %s", cmdstring, strerror(errno)); // 这里务必要把 errno 信息输出或记入 Log
    return XXX;
}

if(WIFEXITED(status))
{
    printf("normal termination, exit status = %d\n", WEXITSTATUS(status)); //取得 cmdstring 执行结果
}
else if(WIFSIGNALED(status))
{
    printf("abnormal termination,signal number =%d\n", WTERMSIG(status)); //如果 cmdstring 被信号中断，取得信号值
}
else if(WIFSTOPPED(status))
{
    printf("process stopped, signal number =%d\n", WSTOPSIG(status)); //如果 cmdstring 被信号暂停执行，取得信号值
}
```

到于取得子进程返回值的相关介绍可以参考另一篇文章：
http://my.oschina.net/renhc/blog/35116

system()函数用起来很容易出错，返回值太多，而且返回值很容易跟 command 的返回值混淆。这里推荐使用 popen()函数替代，关于 popen()函数的简单使用也可以通过上面的链接查看。

popen()函数较于 system()函数的优势在于使用简单，popen()函数只返回两个值：
成功返回子进程的 status，使用 WIFEXITED 相关宏就可以取得 command 的返回结果；
失败返回-1，我们可以使用 perro()函数或 strerror()函数得到有用的错误信息。

这篇文章只涉及了 system()函数的简单使用，还没有谈及 SIGCHLD、SIGINT 和 SIGQUIT 对 system()函数的影响，事实上，之所以今天写这篇文章，是因为项目中因有人使用了 system()函数而造成了很严重的事故。现像是 system()函数执行时会产生一个错误：“No child processes”。

关于这个错误的分析，感兴趣的朋友可以看一下：http://my.oschina.net/renhc/blog/54582

Using console commands in code

/Programming in C/C++ / Console / Using console commands in code
Adam

```
#include <stdlib.h>
    //Console command:
    system("cp file.x newfile.x");

    //Execute file:
    execlp("/usr/bin/omxplayer", " ", "/home/pi/projects/game/audio/alpha.wav", NULL);
        //Execute file: file, arguments (1 or more strings followed by NULL -
```

omxplayer has [OPTIONS] [FILE] hence the blank string)

System Function (Calling shell)

```
#include <stdlib.h>


    system("cp file.x newfile.x");
```

Call On A Background Thread

Can you get away with just adding a '&' to the end of the line to cause it to be done in the background? If not:

```
#include <unistd.h>
    int pid;
    pid=fork();
    if(pid==0)
    {
        //printf("I am the child\n");
        execlp("/usr/bin/omxplayer", " ", "/home/pi/projects/game/audio/alpha.wav",
NULL);      //Execute file: file, arguments (1 or more strings followed by NULL -
omxplayer has [OPTIONS] [FILE] hence the blank string)
        _exit(0);
    }
    else
    {
        //printf("I am the parent\n");
        wait();
    }
```

Running as a different root user

If your application has been run as root user with sudo because you are using the IO pins you may want to make command lines calls as the standard pi user. You can change to a different user using su — USERNAME -c before the command and surrounding it with quotes.

```
    system("su - pi -c \"fetchmail > /dev/null\"");
```

Getting The Result Of An Executed Console Command

```
#include <string>
#include <iostream>
#include <stdio.h>
using namespace std;    //Or use std::string;

//*************************************************************
//*************************************************************
//********** EXECUTE CONSOLE COMMAND AND GET RESULT **********
//*************************************************************
//*************************************************************
//Note, this will return the output stdout. Some commands may generate a stderr
response instead of stdout which you'd see on the console but this won't
//return, unless you simply add " 2>&1" to the end of your command string.  Then
you'll get the output of stdout and stderr
```

```
string do_console_command_get_result (char* command)
{
    FILE* pipe = popen(command, "r");     //Send the command, popen exits immediately
    if (!pipe)
        return "ERROR";

    char buffer[128];
    string result = "";
    while(!feof(pipe))                          //Wait for the output resulting from the
command
    {
        if(fgets(buffer, 128, pipe) != NULL)
            result += buffer;
    }
    pclose(pipe);
    return(result);
}
```