

完成后, 我们使用 quit 命令退出 MySQL 控制台, 并重新以其他用户名 (例如 pi) 登录, 语法如下:

```
$ mysql -u<username> -p
```

注意 SQL 语句中使用单引号是为了避免 MySQL 将字符串理解为保留关键字或已知的名称 (例如表名或者列名) 而出现错误。

插入数据

本节我们将介绍如何将数据插入 MySQL。我们以 pi 用户登录控制台, 准备插入数据。插入命令就是英语单词 INSERT, 它的语法如下所示。

```
INSERT INTO <TABLE> (<FIELD1>, <FIELD2>, ... <FIELDX>) VALUES ('<VAL1>', '<VAL2>', ... '<VALX>');
```

了解了语法, 让我们来写一个完整的 SQL 语句将数据插入数据表。我们通过以下的语句将任务插入 todoitem 表 (在使用 USE pi; 之后)。

```
INSERT INTO todoitem (description, owner, date, location, importance, creator) VALUES
('Finish LAMP Chapter', 'David', '2012-09-22', 'Australia', 'HIGH', 'David');
Query OK, 1 row affected (0.43 sec)
```

MySQL 输出 Query OK 表明插入成功, 让我们在 todoitem 表中插入一条需要 Peter 完成的任务。

```
INSERT INTO todoitem (description, owner, date, location, importance, creator) VALUES
('Finish GUI Chapter', 'Peter', '2012-09-22', 'Hong Kong', 'HIGH', 'David');
Query OK, 1 row affected (0.48 sec)
```

我们成功地插入了两条数据, 那么我们如何确认这些数据真正进入了数据库表, idnumber 是否正确地自动增长了, 这些都可以通过查询语句确定。

查询数据库

查询命令和其他的命令有所不同, 它不使用 QUERY 作为命令 (query 的中文意思就是查询), 而是使用 SELECT, 查询语句的语法如下:

```
SELECT <Fields1>, <Field2>... <FieldX> FROM <TABLENAME> WHERE <INFORMATION QUERY>
```

查询语句的语法有点复杂, 不过当你掌握之后就会了解其中的优点。SELECT 可以查询你指定的列, 也可以使用通配符 * 查询所有的列, 所以我们使用以下语句查询 todoitem 表。

```
mysql> SELECT * FROM todoitem;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| idnumber | description          | owner | date       | location | importance | creator |
+-----+-----+-----+-----+-----+-----+-----+
|      1  | Finish LAMP Chapter | David | 2012-09-22 | Australia | HIGH       | David   |
|      2  | Finish GUI Chapter  | Peter | 2012-09-22 | Hong Kong | HIGH       | David   |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

我们没有使用 **WHERE** 部分，因为我们想查看表中的所有数据，从输出可以看到我们之前插入的两条数据，并且 **idnumber** 也自动增长了。这是最基本的查询语句，下面让我们来做一些复杂的查询，例如我们想查询是谁给 David 创建了任务。首先是要查询出创建者 (**creator**)，我们可以这么写：

```
SELECT creator FROM todolist
```

之后我们指定任务的拥有者 (**owner**) 是 David，我们加上 **WHERE owner = "David"**，最后得到了完整的查询语句。

```
SELECT creator FROM todolist WHERE owner = "David";
+-----+
| creator |
+-----+
| David   |
+-----+
1 row in set (0.00 sec)
```

我们查询出是 David 自己为自己创建的任务。我们再介绍另一个例子，如何查询完成日期在 2012 年 9 月 20 日之后的任务及其重要性 (**importance**)。我们指定查询的列包括描述 (**description**) 和重要性 (**importance**)，并指定日期 (**date**) 迟于 2012-09-20，MySQL 可以直接使用逻辑符号比较日期，例如迟于就是 **>**。我们最后得到的查询语句如下：

```
mysql> SELECT description, importance FROM todolist WHERE date > "2012-09-20";
+-----+-----+
| description          | importance |
+-----+-----+
| Finish LAMP Chapter | HIGH      |
| Finish GUI Chapter  | HIGH      |
+-----+-----+
2 rows in set (0.01 sec)
```

我们最后还要介绍如何在查询语句中使用 **ORDER BY**，**ORDER BY** 可以被追加在查询语句的最后，对查询的结果排序，比如我们希望查询结果按照插入的顺序 (**idnumber**) 显示，我们使用以下语句：

```
mysql> SELECT description, importance, idnumber FROM todolist WHERE date > "2012-09-20" ORDER BY idnumber;
+-----+-----+-----+
| description          | importance | idnumber |
+-----+-----+-----+
| Finish LAMP Chapter | HIGH      | 1        |
| Finish GUI Chapter  | HIGH      | 2        |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

这个查询似乎效果不明显，因为之前没有 **ORDER BY** 也得到了一样的顺序。所以我们将排序的方向调整一下，默认情况下排序采用的是从小到大的顺序 (**ASC**)，我们指定从大到小的顺

序 (DESC)，看看有什么变化。

```
mysql> SELECT description, importance, idnumber FROM todolist WHERE date > "2012-09-20" ORDER BY
idnumber DESC;
+-----+-----+-----+
| description          | importance | idnumber |
+-----+-----+-----+
| Finish GUI Chapter   | HIGH      | 2        |
| Finish LAMP Chapter  | HIGH      | 1        |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

可以看到输出的顺序和之前完全相反了，不过需要注意的是使用排序会导致查询的性能降低，所以请谨慎使用。到这里我们学习了查询语句的主要部分，学习了如何插入和读取数据，下一节我们介绍如何修改数据。

修改数据

这一节我们来修改 (update) 数据，UPDATE 命令有点像 INSERT 和 SELECT 的混合体，因为它同时需要知道调整哪一条数据，并且也需要知道将数据调整成什么样。UPDATE 的语法如下：

```
UPDATE <table name> SET <column name1> = "<value1>", <column name2> = "<value2>"...
<column nameX> = "<valueX>" WHERE <Information Query>
```

我们举个例子，例如我们希望将“Finish LAMP Chapter”这个任务的完成日期 (date) 推后到 23 号，那么我们首先调整 date 列的数据 (先不要执行，我们还没写完)。

```
UPDATE todolist SET date="2012-09-23" WHERE
```

但我们如何区分已有的两条数据呢，它们拥有相同的日期 (date)、重要性 (importance) 和创建者 (creator)，我们可以选择的有 idnumber、拥有者 (owner) 或者描述 (description)。这里我们选择 idnumber，它是主键，可以唯一确定一行数据，于是我们更新之前的 UPDATE 语句。

```
mysql> UPDATE todolist SET date="2012-09-23" WHERE idnumber=1;
Query OK, 1 row affected, 1 warning (0.48 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

这里我们没有使用双引号包围数字，这是因为 MySQL 对待 INT 类型和 VARCHAR 类型是有区别的。不使用双引号的值将直接解析，所以字符串需要引号，而数字不需要。让我们再次确认一下刚才的修改。

```
mysql> SELECT * FROM todolist WHERE idnumber=1;
+-----+-----+-----+-----+-----+-----+-----+
| idnumber | description          | owner | date       | location | importance | creator |
+-----+-----+-----+-----+-----+-----+-----+
| 1        | Finish LAMP Chapter | David | 2012-09-23 | Australia | HIGH      | David   |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

可以看到 UPDATE 的确生效了。我们调整了一章的截止日期，似乎日程有些紧张，所以我们想将所有的任务重要性都提高一些，到 highest（最高），所以我们使用以下语句来调整。

```
UPDATE todolist SET importance="HIGHEST" WHERE
```

WHERE 如何完成呢？因为每个条目的重要性都是 high，所以我们搜索 HIGH，并且更新为 HIGHEST。

```
mysql> UPDATE todolist SET importance="HIGHEST" WHERE importance="HIGH";
Query OK, 2 rows affected (0.49 sec)
Rows matched: 2 Changed: 2 Warnings: 0
```

由于没有输出，我们再查询一下。

```
mysql> SELECT * FROM todolist;
```

idnumber	description	owner	date	location	importance	creator
1	Finish LAMP Chapter	David	2012-09-23	Australia	HIGHEST	David
2	Finish GUI Chapter	Peter	2012-09-22	Hong Kong	HIGHEST	David

```
2 rows in set (0.00 sec)
```

我们成功地将每个任务的重要性从 HIGH 调整到 HIGHEST，完成了修改数据，我们最后介绍如何删除。

删除数据

我们想要在任务完成后删除表中的数据，可以使用 DELETE 命令，该命令语法有些类似于 SELECT。

```
DELETE FROM <table name> WHERE <information query>;
```

我们举个例子，例如我们完成了 LAMP 这一章，于是将其从表中删除。我们选择使用 idnumber 确定该行，命令如下：

```
mysql> DELETE FROM todolist WHERE idnumber=1;
Query OK, 1 row affected (0.42 sec)
```

和 UPDATE 命令一样，DELETE 命令并不显示调整后的表内容，而是提示该命令影响了多少行数据，所以我们再次执行 SELECT 看看删除后的表。

```
mysql> SELECT * FROM todolist;
```

idnumber	description	owner	date	location	importance	creator
2	Finish GUI Chapter	Peter	2012-09-22	Hong Kong	HIGHEST	David

```
1 row in set (0.01 sec)
```

现在表中只剩一行数据了，删除成功。好了，我们来介绍本节最后一个操作命令，删除整张表。删除整张表可以使用 **DROP** 命令，它的语法很简单。

```
DROP TABLE <table name>;
```

让我们把数据库中的唯一一张表删掉吧。

```
mysql> DROP TABLE todolist;
Query OK, 0 rows affected (0.49 sec)
```

成功完成后，我们只剩一个空的 **pi** 数据库了。如果你还想删除数据库，也是使用 **DROP** 命令，将 **TABLE** 换成 **DATABASE** 并加上数据库的名称即可。介绍完 MySQL 数据库，我们下一节将要学习 LAMP 的最后一个部分：PHP。

8.5 P: PHP

PHP 是一套可以动态生成网页的系统。起初，所有的网页都只使用 **HTML** 来生成，这使得网页无法有效地处理用户的输入和动作。为了解决 **HTML** 只能生成静态网页的问题，人们发明了 **CGI** (Common Gateway Interface)。**CGI** 使 Web 服务器具有了按需提供网页内容的能力。

早期的 **CGI** 程序会根据输入信息输出各种 **HTML** 片段，再将大量重复片段组合成完整的 **HTML** 文件，这其中不可避免地有各种无用功。而 **PHP** 可以将动态生成的代码片段嵌入静态 **HTML** 文件中，高效地实现与 **CGI** 相同的效果。

从诞生之日起，**PHP** 就以星火燎原之势迅速风靡世界，下载安装量已超过两千万。获得这样的成功主要是因为 **PHP** 可以用比较小的开发成本实现与 **CGI** 相同的功能。现在让我们使用 **apt-get** 命令把 **PHP** 安装到树莓派上。

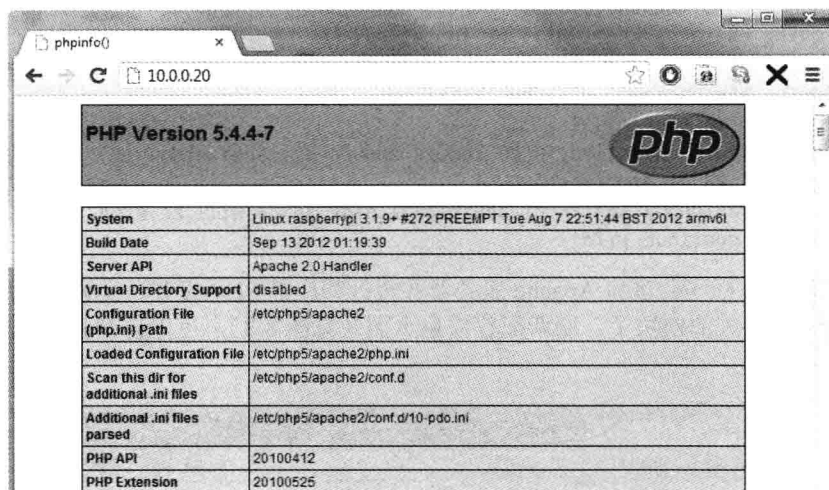
```
$ sudo apt-get install php5
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5-cli php5-common
Suggested packages:
  php-pear
The following packages will be REMOVED:
  apache2-mpm-worker
The following NEW packages will be installed:
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5 php5-cli php5-common
0 upgraded, 7 newly installed, 1 to remove and 84 not upgraded.
Need to get 5,707 kB of archives.
After this operation, 16.3 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

从安装信息中可以看到，与 **PHP** 一起安装的还有相应的 **Apache** 适配模块。安装完成后，首先我们来测试下它是否能够正常工作。这里我们来写一个简单的 **PHP** 页面，它能够告知我们 **PHP**

的安装信息和运行情况。我们需要将/var/www 文件夹下的 index.html 文件替换为包含下面这行内容的 index.php 文件。

```
<?php phpinfo(); ?>
```

删除 index.html 文件并添加 index.php 文件后, 用浏览器打开 Apache 服务器的默认起始页面就可以看到图 8-2 所示的结果。



PHP Version 5.4.4-7	
System	Linux raspberrypi 3.1.9+ #272 PREEMPT Tue Aug 7 22:51:44 BST 2012 armv6l
Build Date	Sep 13 2012 01:19:39
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini
PHP API	20100412
PHP Extension	20100525

图 8-2 PHP 信息页面

在这个页面上可以看到所有我们新安装的这个 PHP 实例的配置信息。同时这个页面的成功打开也意味着 PHP 解释器工作正常, 以及 index.php 成功被 Apache 服务器识别为默认起始页面。生成页面的这行代码是一个很好的 PHP 程序范例。

```
<?php phpinfo(); ?>
```

首先代码主体被<?php 和?>包含, 表示内容将由 PHP 解释器来解析, 这里只涉及了一个函数: phpinfo(), 用来显示 PHP 实例的配置信息。

这里提到的函数是指通过特定的名字(函数名)来调用一段现有代码的方式。函数名后面总会跟一对圆括号, 用来向函数传递参数。PHP 也允许使用数组进行参数传递, 关于数组的内容在第 7 章我们已经进行了详细的讲述。这里只略作回顾。数组是一种用来存储变量的变量, 既可以对整个数组进行赋值, 也可以对数组中的单个元素进行赋值。我们可以通过在数组名后面加特定数字的方式访问特定元素(例如: array[6]), 注意数组中第一个元素的标记为 0(例如 array[0])。

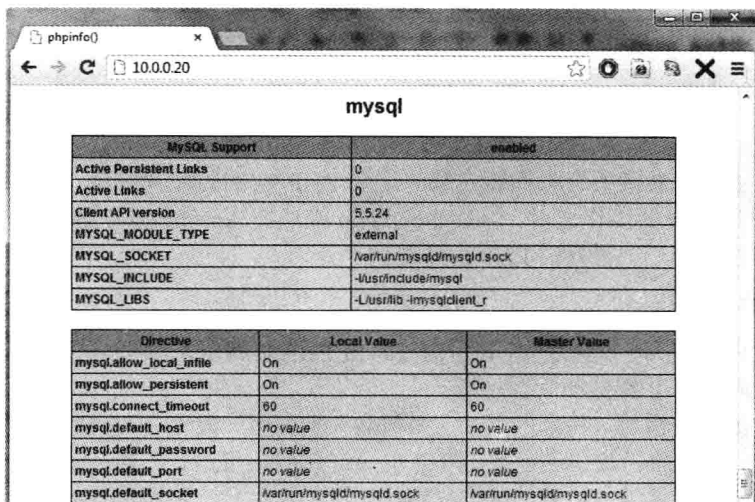
关于 PHP 我们还应该了解以下这些内容。

- PHP 中的声明语句需要以分号结束。
- PHP 中的变量需要以美元符号 (\$) 作为前缀(这一点与 BASH 相同)。
- PHP 中的数组需要 @ 符号作为前缀。
- PHP 中的函数主体通过一对大括号进行界定 ({ })。

下面我们来看下这张配置信息页面上的内容，从 MySQL 部分开始。什么，我们还没提过 MySQL 设置？是这样的，PHP 需要通过和 MySQL 通信来在网页上显示数据库中的内容。好在通过 apt-get 就可以简单地进行设置。

```
$ sudo apt-get install php5-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libmysqlclient18
The following NEW packages will be installed:
  libmysqlclient18 php5-mysql
0 upgraded, 2 newly installed, 0 to remove and 84 not upgraded.
Need to get 711 kB of archives.
After this operation, 3,547 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

在安装过程中，程序会重启 Apache 服务器并进行相应的 PHP 设置。安装成功后，我们就可以使用与 MySQL 相关的模块了。刷新配置信息页面（见图 8-3），可以看到我们的 MySQL 已经配置好了。



MySQL Support		enabled
Active Persistent Links	0	
Active Links	0	
Client API version	5.5.24	
MYSQL_MODULE_TYPE	external	
MYSQL_SOCKET	/var/run/mysqld/mysqld.sock	
MYSQL_INCLUDE	-I/usr/include/mysql	
MYSQL_LIBS	-L/usr/lib -mysqlclient_r	

Directive	Local Value	Master Value
mysql.allow_local_infile	On	On
mysql.allow_persistent	On	On
mysql.connect_timeout	60	60
mysql.default_host	no value	no value
mysql.default_password	no value	no value
mysql.default_port	no value	no value
mysql.default_socket	/var/run/mysqld/mysqld.sock	/var/run/mysqld/mysqld.sock

图 8-3 PHP MySQL 配置信息

现在我们已经有了：

- 可以使用的 PHP；
- 工作正常的 Apache 服务器；
- 可以与 PHP 通信的 MySQL。

这意味着我们的 LAMP 系统已经整装待发了！

8.6 实现一个简易网页应用

在树莓派上运行一套 LAMP 系统意味着我们可以在上面随心所欲地测试各种 Web 应用。在没有 PHP 甚至没有 HTML 基础时，开发 Web 应用听起来异常艰难。别害怕，在这一点上 Web 开发与 Linux 有些类似，初看上去有些令人生畏，实际上没有什么不可克服的困难，你需要的不过是一点指导而已。实践出真知，这总是没有错的，下面我们就来用 LAMP 系统创建一个 Web 应用。

要创建 Web 应用，我们需要了解一些 HTML 和 PHP 知识。靠它们构建起网页骨架后，页面看起来会有些简陋，好在有很多方法可以进行网页美化，这方面的书并不难找。

HTML

超文本标记语言（HTML）是构建互联网的基石。HTML 通过一系列的标签实现了承载 Web 应用内容的基础。在收到 HTML 文本后，浏览器根据这些标签渲染生成网页。这样的工作原理带来了低流量消耗和良好的跨平台性等好处，缺点是平台不同看到的网页也会有些许不同。

HTML 标签由中括号包含，通常两个一组，偶尔有单个出现的情况，例如：

- `<html></html>`：这组标签指明 HTML 页面开始和结束的位置。
- `<hr>`：这一标签单独出现，其作用是在页面上显示一条横线。

`</html>`中 `html` 之前的反斜杠表示这组标签的结束，`<hr>`中 `hr` 之后的反斜杠表示这一标签单独出现。好了，现在来看一个简单的网页吧。

```
<html>
  <head>
    <title>Pi Brand - Todo List App</title>
  </head>
  <body>
    <h1>Pi Todo List App</h1>
  </body>
</html>
```

将上面这段代码保存到 `/var/www` 文件夹下的 `index.php` 文件中，在浏览器中刷新我们刚才打开的网页，就可以看到一个内容和标题都为“Pi Todo List App”的页面。对照这个例子我们来解析下 HTML 文件的结构。

- 首先，所有的内容都包含在一组 `<html></html>` 标签之间。
- 第二级，有 `head` 和 `body` 这两个标签，分别包含页面的页首信息和实际内容。
- `title` 标签用于设置标签栏信息。
- `h1` 标签（`head1`）表示最上级的标题，使用所有标题类标签中最大的字体来显示。通过 `hx`（`x` 表示任意数字）这样的形式可以定义更多的标题级别，多数浏览器支持 4~5 级标题。

上面这个例子中涉及的 5 种 HTML 标签都是最为常见的。实际上 HTML 标签的种类非常多，wc3schools.com 这个网站上的“Beginning HTML with CSS and XHTML”（Craig Cook and David Schultz）这一教程值得参考。下面罗列一些常用的 HTML 标签。

- `<p>`: Paragraph 标签, 用于标记文本段落。
- `<br>`: Break 标签, 用于插入新的一行。
- `<hr>`: Horizontal 标签, 用于在页面上显示一条横线。
- `<a>`: Hyperlink 标签, 用于标记超链接。
- `<img>`: Image 标签, 用于标记图片。

表格类标签:

- `<table>`: 表格标签。
- `<th>`: 用于标记表头单元格。
- `<tr>`: 用于标记表格中的一行。
- `<td>`: 用于标记单元格。

表单类标签:

- `<form>`: 表单标签。
- `<input>`: 用于标记表单中的输入框。

这些标签后面我们会逐步涉及, 让我们再来看看 PHP 相关的内容。

开始构建网页

我们将要构建的应用由两部分构成, 一个和数据库连接的待办事项列表和一个用于添加新待办事项的输入表单。此外, 我们还需要通过复选框来选择要删除的条目。有了基本页面结构后我们再来构建页面逻辑。首先, 我们需要从数据库里读取现有的信息。前文中我们已经完成了数据库设定以及相应数据库和表的创建, 这里直接使用就好。下面是创建 `todolist` 表的 `CREATE` 语句。

```
create table todolist (
  idnumber INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  description VARCHAR(200),
  owner VARCHAR(40),
  date DATE,
  location VARCHAR(40),
  importance VARCHAR(10),
  creator VARCHAR(40)
);
```

有了网页结构和数据库后我们要关心的就是如何显示数据了。我们要显示的数据非常结构化, 所以我们可以通过表格来组织这些数据。通过 `mysql` 接口我们可以实现 MySQL 和 PHP 间的连接。

```
<?php
$mysqli = new mysqli('localhost', 'pi', 'raspberrypi', 'pi');
if ($mysqli->connect_error) {
    die('Connect Error (' . $mysqli->connect_errno . ') '
        . $mysqli->connect_error);
}
```

```
$mysqli->close();
?>
```

这段代码的作用有：

- 创建一个新的 `mysqli` 对象来连接数据库，服务器为本机，用户名为 `pi`，密码为 `raspberry`。
- 检查连接是否成功，如果不成功则显示相应的错误信息。
- 在使用完毕后断开数据库连接。

将这段连接数据库的代码添加到我们原有的代码中。

```
<html>
<head>
<title>Pi Brand - Todo List App</title>
</head>
<body>
<h1>Pi Todo List App</h1>
<?php
$mysqli = new mysqli('localhost', 'pi', 'raspberry', 'pi');
if ($mysqli->connect_error) {
    die('Connect Error (' . $mysqli->connect_errno . ') '
        . $mysqli->connect_error);
}
$mysqli->close();
?>
</body>
</html>
```

数据库内容显示

需要显示的内容由两部分构成，即表头和从数据库取出的实际内容。我们可以通过一个循环来获取每一行需要显示的内容并加上相应的 HTML 标签。表格的基本结构如下：

```
<table>
<tr>
<th>Description</th>
<th>Owner</th>
<th>Due Date</th>
<th>Location</th>
<th>Importance</th>
<th>Creator</th>
</tr>
...
</table>
```

通过下面的 PHP 代码我们可以生成需要显示的行。

```
<?php
$result = $mysqli->query("SELECT * FROM todolist");
```

```

while($row = $result->fetch_assoc()){
    print "<tr>";
    print "<td>".$row["description"]."</td>";
    print "<td>".$row["owner"]."</td>";
    print "<td>".$row["date"]."</td>";
    print "<td>".$row["location"]."</td>";
    print "<td>".$row["importance"]."</td>";
    print "<td>".$row["creator"]."</td>";
    print "</tr>";
}
?>

```

首先声明变量\$result，用于存放查询数据库后得到的查询结果。这里要用到的 SQL 语句显然是 SELECT * FROM todolist。查询返回内容由\$result->fetch_assoc()语句逐行传递给 while 循环中的变量\$row，经过格式整理后最终输出。可以看到输出中混合了 PHP 变量和 HTML 标签，这是为了和前面的表头保持格式一致。

再将这部分代码加入原有的代码段中。此外，我们还需要将\$mysqli->close(); 语句移到 PHP 代码段的最后，以保证查询完成后才断开数据库连接。

```

...
<body>
<h1>Pi Todo List App</h1>
<?php
$mysqli = new mysqli('localhost', 'pi', 'raspberry', 'pi');
if ($mysqli->connect_error) {
    die('Connect Error (' . $mysqli->connect_errno . ') '
        . $mysqli->connect_error);
}
?>
<table>
<tr>
<th>Description</th>
<th>Owner</th>
<th>Due Date</th>
<th>Location</th>
<th>Importance</th>
<th>Creator</th>
</tr>
<?php
$result = $mysqli->query("SELECT * FROM todolist");
while($row = $result->fetch_assoc()){
    print "<tr>";
    print "<td>".$row["description"]."</td>";
    print "<td>".$row["owner"]."</td>";
    print "<td>".$row["date"]."</td>";
    print "<td>".$row["location"]."</td>";
    print "<td>".$row["importance"]."</td>";
    print "<td>".$row["creator"]."</td>";

```

```

        print "</tr>";
    }
    $mysqli->close();
?>
</table>
</body>
</html>

```

网站数据插入

至此，HTML 和 PHP 已经可以配合工作了，HTML 提供网页框架，PHP 则负责查询数据库以及为网页填充内容。为了能够添加新待办事项，我们需要一个输入表单。新待办事项的各种构成元素都应该有对应的输入框。我们还需要一个隐藏元素并声明一个变量 `action` 来定义我们要对数据进行的操作。

```

...
</table>
<form action="index.php" method="POST">
<input type="hidden" name="action" value="insert" />
Description: <input name="description" /><br/>
Owner: <input name="owner" /><br/>
Date: <input name="date" /><br/>
Location: <input name="location" /><br/>
Importance: <input name="importance" /><br/>
Creator: <input name="creator" /><br/>
<input type="submit" />
</form>
</body>
</html>

```

将加粗的这段代码添加到生成表格代码之后，不妨添加一个 `<hr/>` 标签将它们隔开。显示效果如图 8-4 所示。



Pi Brand - Todo List App

10.0.0.20

Pi Todo List App

Description	Owner	Due Date	Location	Importance	Creator
Finish LAMP Chapter	David	2012-09-22	Australia	HIGH	David

Description:

Owner:

Date:

Location:

Importance:

Creator:

图 8-4 添加输入表单后的页面

如果这时我们按下 Submit 按钮，会发现页面没有发生变化，但是 URL 不同了。这是 CGI 命令的执行方式，现在我们就开始向 PHP 代码中添加 CGI 处理逻辑。

上面我们使用了 POST 方法来发送请求，多数情况下 POST 方法和 GET 方法的功能是一致的。唯一的不同在于 GET 方法会将数据内容显示在 URL 中，而 POST 方法不会。如果在上面的例子中使用 GET 方法，按下 Submit 按钮后 URL 会变成如下的形式：`http://10.0.0.20/index.php?action=insert&description=&owner=&date=&location=&importance=&creator=`。

通常会有大量的数据需要进行传输和处理，好在 PHP 大大简化了这一过程。它提供了 `_POST`、`_GET` 以及 `_REQUEST` 这 3 种特殊变量。

首先，我们要检查变量 `action` 是否已赋值。若 `action` 已赋值则根据不同的值进行不同的数据库操作，代码如下：

```
if(isset($_REQUEST["action"])){
    switch($_REQUEST["action"]){
        case "insert":
            $SQL="INSERT INTO todolist (description, owner, date, location,
importance, creator)
VALUES (";
                $SQL=$SQL."".$_REQUEST["description"]."',";
                $SQL=$SQL."".$_REQUEST["owner"]."',";
                $SQL=$SQL."".$_REQUEST["date"]."',";
                $SQL=$SQL."".$_REQUEST["location"]."',";
                $SQL=$SQL."".$_REQUEST["importance"]."',";
                $SQL=$SQL."".$_REQUEST["creator"]."'"';
                $SQL=$SQL.");";
            if ($mysqli->query($SQL)=== FALSE) {
                printf("Error - Unable to insert data to table " . $mysqli->error);
            }

            break;
        case "delete":
            print "Delete function yet to be added!";
            break;
    }
}
```

代码中的 switch 语句主要针对变量 `action` 的不同值来执行相应的操作。这里我们添加了 insert 和 delete 两个分支，delete 分支稍后我们会做进一步修改。

将新待办事项插入数据库

我们需要构建简单的 SQL 语句来完成插入的操作。与前文讲解过的插入语句相同，我们需要使用 INSERT 关键字，指出表明和需要写入的字段，然后逐个变量进行字段拼接。每个字段的前后都要用单引号包含，其后还要添加逗号，使用双引号转义后一个字段看起来像这样：

```
$SQL=$SQL."".$_REQUEST["description"]."',";
```

这条语句表示将\$_REQUEST["description"]添加到变量\$SQL 中。

■ **注意** 这样的插入方式虽然简便，却存在安全隐患。进行插入内容检测是必要的，推荐使用 `mysqli_real_escape_string`¹ 来完成检测。

变量 SQL 构造好之后，我们就可以通过 `mysqli` 查询接口将相应内容插入数据库。在前面的代码中我们甚至还添加了查询异常检测。现在，整个网页会按照这个顺序载入：显示页首信息，创建数据库连接，执行 CGI 逻辑，显示现有待办事项，最后显示输入表单。将数据库插入这部分代码添加到原有的代码段后，我们就可以尝试在输入表单中填写内容来进行测试。单击 Submit 按钮后，我们会看到表格中出现新的一列（见图 8-5）。



图 8-5 作者的待办事项

删除待办事项

搞定了添加功能，我们再来看看如何删除。虽说条条大路通罗马，但路途远近难易各有不同。想要执行删除操作大致有两种方法。第一种最为简单，在表格中的每一行后面添加独立的删除选项；第二种，在每行前面添加复选框，这样就可以方便地执行批量删除操作。显然，第二种方法更加灵活方便，下面我们将重点讨论它。

要实现删除功能，我们需要在原有的代码中做出两处修改。第一，在表格外层添加删除表单，将变量 `action` 赋值为 `delete`，以及在表单下方添加 Submit 按钮；第二，在每行前面添加复选框。修改完成后将如图 8-6 所示。

具体的实现代码如下所示。可以看到表格外层的删除表单和我们前面提到的插入表单看起来十分相似，只是变量 `action` 被赋值为 `delete`。最为重要的变化是我们在 `while` 循环中添加了生成复选框的语句，复选框的值 `idnumber` 可以通过语句中定义的数组进行操作。

¹ 译者注：详细使用方法可参见 http://www.w3school.com.cn/php/func_mysqli_real_escape_string.asp。

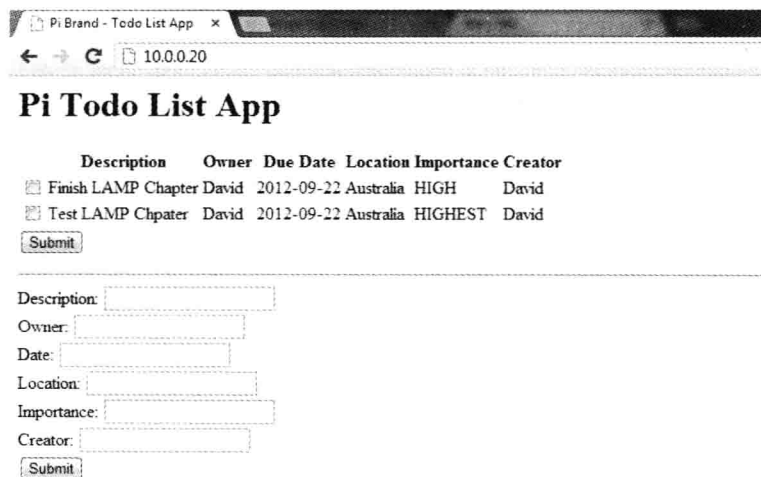


图 8-6 添加复选框后

加粗的部分是我们新添加的代码。

```
...
<h1>Pi Todo List App</h1>
<form action="index.php" method="POST">
<input type="hidden" name="action" value="delete" />
<table>
<tr>
<td></td>
...
</tr>
<?php
$result = $mysqli->query("SELECT * FROM todolist");
while($row= $result->fetch_assoc()){
    print "<tr>";
    print "<td><input type='checkbox' name='checkboxes[]' value='".$row["idnumber"].
    "' /></td>";
    print "<td>".$row["description"]."</td>";
    print "<td>".$row["owner"]."</td>";
    print "<td>".$row["date"]."</td>";
    print "<td>".$row["location"]."</td>";
    print "<td>".$row["importance"]."</td>";
    print "<td>".$row["creator"]."</td>";
    print "</tr>";
}
$mysqli->close();
?>
</table>
<input type="submit"/>
</form>
...
```

从数据库删除数据

最后，我们需要添加 `delete` 操作的处理逻辑。与 `insert` 操作的处理方式有些类似，我们需要使用 `DELETE` 关键字，指出需要删除的条目的 `idnumber`，然后逐一进行字段拼接。这里使用 `for` 循环实现。

```
for($i=0; $i < count($_REQUEST['checkboxes']); $i++){
```

在这个 `for` 循环中，我们需要依次读取数组 `$_REQUEST['checkboxes']` 中第 `i` 个元素的值，并在每个值后面拼接字符串 `"or"`，以达到同时删除多个条目的目的。在字符串拼接完成后会发现在末尾处多加了一个 `"or"`，我们可以通过 `rtrim` 函数来删除这个多余的 `"or"`。这样我们就可以向数据库发送请求了，实际代码段如下：

```
$SQL="DELETE FROM todolist WHERE";
for($i=0; $i < count($_REQUEST['checkboxes']); $i++){
    $SQL=$SQL . " idnumber=" . $_REQUEST['checkboxes'][$i] . " or";
}
$SQL= rtrim($SQL, "or");
if ($mysqli->query($SQL)== FALSE) {
    printf("Error Unable to delete value " . $mysqli->error);
}
```

从图 8-7 中我们可以看到，我们删除了前文中插入的一个待办事项。本书上的所有代码都可以在 Apress 的网站¹上下载到，如果对代码的结构有疑问的话，不妨下载一份作为参考。

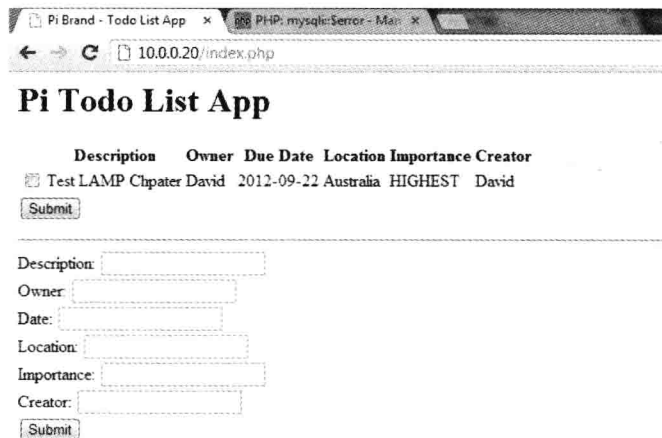


图 8-7 删除一个待办事项后

排错

前面我们已经接触了一些排错的方法。首先，我们应该逐步加入新的代码，并在加入后刷新

¹ 译者注：<http://www.apress.com/9781430248217>。

页面。我们也可以通过浏览器的“查看网页源代码”功能来查看 PHP 所生成的 HTML 代码，来确定错误的位置。如果连 HTML 代码都无法正常生成，我们就需要查看 Apache 的出错信息了。记得检查代码中的引号以及各种括号是否已经成对匹配，switch 语句的分支判断条件后面是否忘记了加分号。

不断试错是学习 PHP 以及 Web 开发的必经之路。我们应该养成自觉仔细检查代码的良好习惯。借助 print 函数，我们可以将一些中间变量值打印出来，帮助进行错误定位。最后，要记得你拥有树莓派上的所有权限，不要害怕出错，要勇于尝试。

开发路线图

调用各种工具，经历了艰苦卓绝地开发，我们的应用已经可以显示、添加以及删除待办事项了。那么如果继续这一应用的开发我们还可以做些什么呢？这里罗列了一些可能的改进，供感兴趣的读者参考。

- 每次插入时调用 isset() 以保证不会插入空白条目。
- 每次插入时调用 mysqli_real_escape_string 函数，避免不安全的输入内容。
- 为两个“Submit”的 value 选项分别赋值，以示区分。
- 尝试使用 CSS 来进行页面美化，不妨查看下 Bootstrap，它是一个不错的起点。

本章小结

在这一章里我们学习了很多新的内容。我们安装配置了 Apache、MySQL 以及 PHP。我们学习了各种基础的数据库操作：数据库的创建和删除、表的创建和删除，以及如何在表中插入、删除数据。我们也了解了 Apache 服务器显示网页的方法。我们甚至还综合以上所有内容，基于 PHP 以及 HTML 实现了一个可以和数据库通信的网页待办事项应用。

这个应用看起来或许有些简陋，却极具里程碑意义。在安装 3 种开发工具，涉猎了 3 种不同的编程语言后，我们成功搭建了自己的应用。这个过程无疑值得纪念，我们不妨举杯相庆。

无线树莓派

“能放进口袋里的电脑”，这正是树莓派的设计初衷之一。外形小巧的优势，大大增加了它的应用场景。通过添加合适的外设，树莓派几乎无所不能。此外它低功耗的特点也降低了对电源的需求。一句话：身材小，本领大！

不幸的是各种线缆束缚了树莓派这一优点的发挥。这些线缆中 5V 的输入电源自然是必不可少的，但树莓派只有在连接其他设备的情况下才能发挥它的作用。如果想把树莓派作为家用电脑使用，则至少需要下面几种组件。

- HDMI 或复合视频输出端子等线缆，用于视频输出。
- 以太网线，用于网络访问。
- 键盘，用于数据输入。
- SD 卡，用于数据存储。

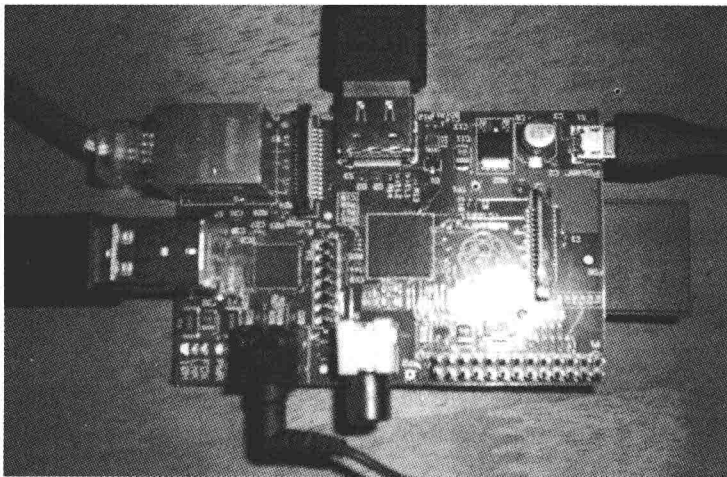


图 9-1 被各种线缆束缚的树莓派

看到这份列表，不难发现，这些插在树莓派上的线缆其实主要都是用来和用户进行基本的交互的。将树莓派从这些线缆的束缚中解放出来，给它插上 WiFi 的翅膀，这正是本章要讨论的主题。

9.1 一点准备工作

为了避免不必要的问题，我们需要保证在树莓派上运行的 Raspbian 是最新版的。

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

执行第一行命令后，apt-get 工具会从软件仓库获取各种最新版的软件信息。执行第二行命令后，apt-get 工具会将本地已安装的软件更新到最新版。

9.2 快刀斩乱麻

在快刀斩乱麻、去除各种线缆前，我们需要先想清楚树莓派的使用场景。不同的使用场景，可以去除的线缆各不相同。树莓派的使用场景多种多样，我们很难一一覆盖。授人以鱼不如授人以渔，本章我们想要教会读者的是确定并移除非必要线缆的技能。

移除输入/输出线缆

柿子要捡软的捏，这里先从最容易移除的线缆着手，即视频输出线、键盘连线以及音频线。这听起来可能有些奇怪，这些线缆听起来似乎是必不可少的。这个问题的本质在于如何不通过这些线缆也能实现树莓派和用户之间的交互。其实在本书中已经介绍了不少远程操作树莓派的方法，在前面的章节中对 SSH 已经进行了比较详细的讨论。远程管理会带来一些额外的好处，如更富裕的内存（通过减少共享显存的配额）。我们可以通过 `sudo raspi-config` 命令将 `memory_split` 设为 240/16。

当然在进行系统安装配置时还是少不了使用显示器和键盘的。一旦完成了这些操作，我们就可以着手移除相应的线缆，进行远程管理了。这意味着你可以通过任意联网的电脑甚至智能手机管理树莓派。

架设远程桌面服务

虽然使用命令行操作好处多多，但不可否认的是不少人还是习惯使用图形界面。此外，少数情况下我们确实必须借助图形界面进行管理。这时我们就需要使用 VNC（Virtual Network Computing）远程桌面。

首先，我们需要在树莓派上安装 VNC 服务器，在自己的电脑上安装 VNC 客户端。VNC 客户端有很多选择，这里使用轻量级的 TightVNC（可从 www.tightvnc.com 下载）。接下来我们在树莓派上安装相应的 VNC 服务器。

```
$ sudo apt-get install tightvncserver
```

你会看到与下面所示类似的内容。

```

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  x11-xserver-utils xfonts-base xfonts-encodings xfonts-utils
Suggested packages:
  tightvnc-java nickle cairo-1.5c xorg-docs-core
The following NEW packages will be installed:
  tightvncserver x11-xserver-utils xfonts-base xfonts-encodings xfonts-utils
0 upgraded, 5 newly installed, 0 to remove and 91 not upgraded.
Need to get 7,824 kB of archives.
After this operation, 11.7 MB of additional disk space will be used.
Do you want to continue [Y/n]?

```

成功安装后，使用下面的命令启动服务器。

```
$ tightvncserver
```

程序会提示你设置两种密码，一种是具有实际操作权限的远程登录密码，另一种只具有查看权限的远程登录密码。

```

You will require a password to access your desktops.
Password:
Warning: password truncated to the length of 8.
Verify:
Would you like to enter a view-only password (y/n)? n
New 'X' desktop is raspberrypi:1
Creating default startup script /home/pi/.vnc/xstartup
Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log

```

完成服务器配置后就可以在自己的电脑上打开 VNC 客户端进行连接了，如图 9-2 所示。

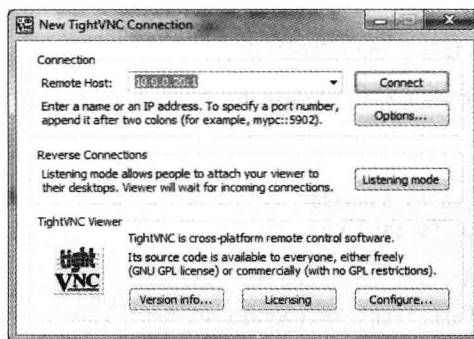


图 9-2 TightVNC 客户端

从图中可以看到在树莓派的 IP 地址后面加上了“:1”。它不是端口号，而是屏幕编号。在上面的 tightvncserver 配置信息中可以看到新创建的远程桌面叫做“raspberrypi:1”。现在单击 Connect 按钮就可以以图形界面访问树莓派了。

我们还可以对 VNC 服务器提供的图像质量进行定制，例如：

```
$ vncserver :1 -geometry 1920x1080 -depth 24
```

这表示 vncserver 运行在 1 号屏幕，分辨率为 1920 像素 x 1080 像素（树莓派所能支持的最大分辨率），色彩深度为 24 位。在实际使用时可以根据网络情况降低一些参数。每次开机都手动启动 vncserver 显然不是个好主意，好在我们可以将它设置为开机自动启动。最简单的方法是修改 /etc/rc.local 文件。

默认情况下该文件内容如下：

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

exit 0
```

可以在 exit 0 前面新插入一行，写入刚才讲解过的命令。

```
vncserver :1 -geometry 1920x1080 -depth 24
```

也可以创建如下脚本，并将其存为 /etc/init.d/tightvnc，更方便地启动和停止 VNC 服务。

```
### BEGIN INIT INFO
# Provides: tightvnc
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start Tight VNC Server at boot time
# Description: Start Tight VNC Server at boot time.
### END INIT INFO
#!/bin/sh
# /etc/init.d/tightvnc
```

```
USER=pi
```

```

HOME=/home/pi
export USER HOME

case "$1" in
    start)
        echo "Starting Tight VNC Server"
        /usr/bin/vncserver :1 -geometry 1920x1080 -depth 24
        ;;

    stop)
        echo "Stopping Tight VNC Server"
        /usr/bin/vncserver -kill :1
        ;;

    *)
        echo "Usage: /etc/init.d/tightvnc {start|stop}"
        exit 1
        ;;
esac

exit 0

```

我们需要为这个脚本文件添加执行权限。

```
$ sudo chmod +x /etc/init.d/tightvnc
```

这样就可以通过执行 `/etc/init.d/tightvnc` 来启动和停止 VNC 服务了。需要注意，该脚本需要使用 `sudo` 执行，首次运行脚本时需要设置 VNC 密码。我们也可以使用这个脚本来实现 VNC 服务的开机启动。

```
$ sudo update-rc.d tightvnc defaults:
update-rc.d: using dependency based boot sequencing
```

这样我们就了解了移除用户输入线缆的方法，下一步是什么呢？

9.3 为树莓派添加 WiFi

可靠的网络对实现远程管理至关重要。现在无线网络的普及程度已经非常高了，这里我们尝试为树莓派配置无线网络。有点遗憾的是树莓派并没有板载无线网卡，好在合适的 USB 无线网卡并不难找。

需要注意的是，由于树莓派的电源功率有限，我们最好选购低功耗的无线网卡。图 9-3 所示的是我们使用的无线网卡（D-Link DWA-131）。

 **注意** 将无线网卡连接到树莓派可能导致重启，请预先保存正在编辑的文件。

它显然不是唯一的选择，大多数无线网卡的配置方法十分类似，当然也可以使用手边闲置的低功耗无线网卡。从图 9-4 中可以看到，我们已经移除了 3 条线缆，以太网线是我们现在的目标。

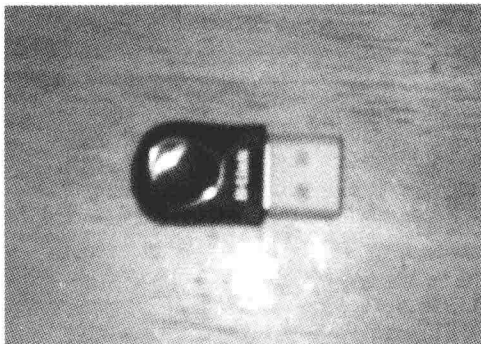


图 9-3 D-Link DWA-131 无线网卡

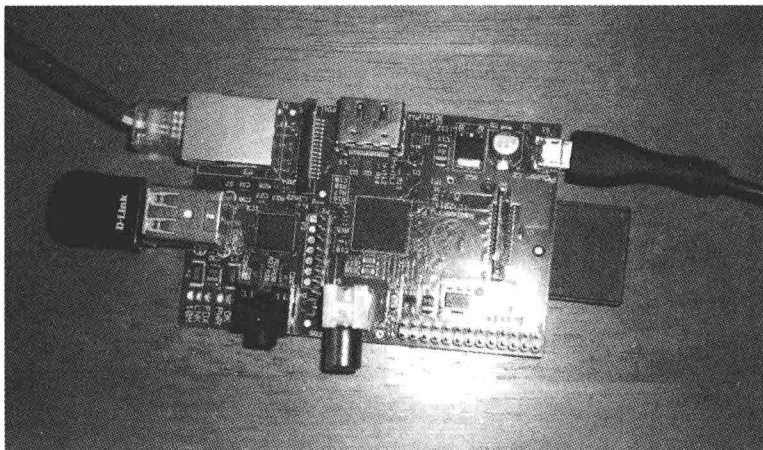


图 9-4 只剩两条线缆喽

下面我们将分别讨论如何使用命令行或是图形界面配置 WiFi。首先通过 `lsusb` 命令来确认系统是否已经识别了无线网卡。

```
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 07d1:3303 D-Link System DWA-131 802.11n Wireless N Nano Adapter
(rev.A1)
[Realtek RTL8192SU]
```

从结果里看到，网卡已经被成功识别了。如果没有识别出网卡也别害怕，我们要做的只是安装相应的网卡驱动。通过搜索引擎不难确认网卡所用的芯片，然后我们只需要搜索并安装相应芯片对应的驱动。成功安装驱动后，再执行 `lsusb` 命令就可以看到无线网卡了。

接下来通过 `ifconfig`（Linux 中最常用的网络管理命令）来确认无线网卡的工作状态。

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:8a:46:ba
          inet addr:10.0.0.20  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6558 errors:0 dropped:0 overruns:0 frame:0
          TX packets:268 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:374403 (365.6 KiB)  TX bytes:28129 (27.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 90:94:e4:51:81:7a
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

3 个段落分别对应了 3 种网络设备。

- eth0 表示以太网卡。
- lo 表示本地回环。
- wlan0 表示无线网卡。与前两种设备不同，这一段中没有 IP 地址以及子网掩码，这是由于无线网卡的相应配置还没有完成。

如果通过 lsusb 命令已经可以确认网卡，但执行 ifconfig 却看不到对应信息，可以尝试重启树莓派。如果问题依然存在，可以考虑重新安装网卡驱动。

我们首先来看如何在图形界面中设置无线网卡。有点讽刺的是，我们需要使用命令行来安装需要用到的图形配置工具 wpaui。

```
$ sudo apt-get install wpaui

Reading package lists... Done
Building dependency tree
Reading state information... Done
wpaui is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

如果使用的是最新版的 Raspbian，它已经预装了 wpa-gui。你会看到如下结果。

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

The following extra packages will be installed:

libaudio2 liblcms1 libmng1 libqt4-svg libqtcore4 libqtgui4 menu

Suggested packages:

nas liblcms-utils qt4-qtconfig menu-l10n

The following NEW packages will be installed:

libaudio2 liblcms1 libmng1 libqt4-svg libqtcore4 libqtgui4 menu wpagui

0 upgraded, 8 newly installed, 0 to remove and 96 not upgraded.

Need to get 8,921 kB of archives.

After this operation, 22.6 MB of additional disk space will be used.

Do you want to continue [Y/n]?

完成安装后，重启树莓派可以避免不必要的错误。

注意 如果你发现无线网卡频繁出现速度缓慢或连接失败，这可能是由于其他 USB 设备，诸如键盘鼠标占用了过多的电源。移除线缆，通过 VNC 或 SSH 访问往往可以解决问题。

在图形化桌面中配置 WiFi

登录树莓派的图形化桌面，应该会出现一个新的 WiFi Config 图标，如图 9-5 所示。

双击该图标将启动名为 `wpa_gui` 的应用程序，界面如图 9-6 所示。



图 9-5 WiFi Config 图标

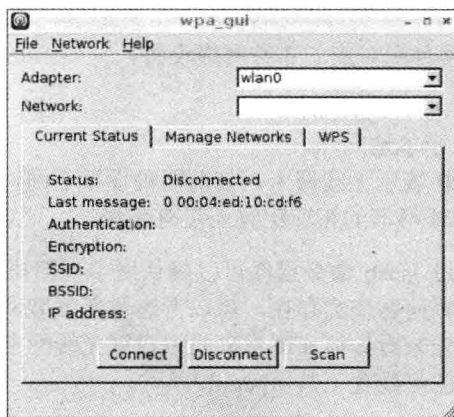


图 9-6 wpa_gui 的界面

在主界面上单击扫描（Scan）按钮，过一会儿将会弹出如图 9-7 所示的窗口。

该窗口显示了搜索到的所有无线网络，选择你自己的无线网络并双击打开配置窗口（见图 9-8）。

在该窗口中填写你自己的无线网络配置，这里有一些需要注意的问题。

- SSID 是你的无线网络名称，请填写正确。
- 在验证方式（Authentication）中填写无线网络的认证方式，可选的有 WEP、WPA、WPA2 等。

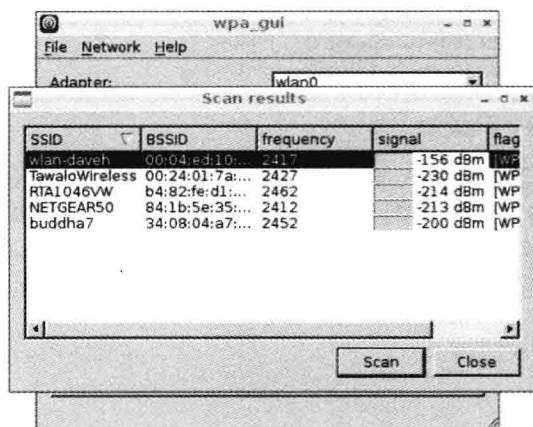


图 9-7 搜索结果窗口

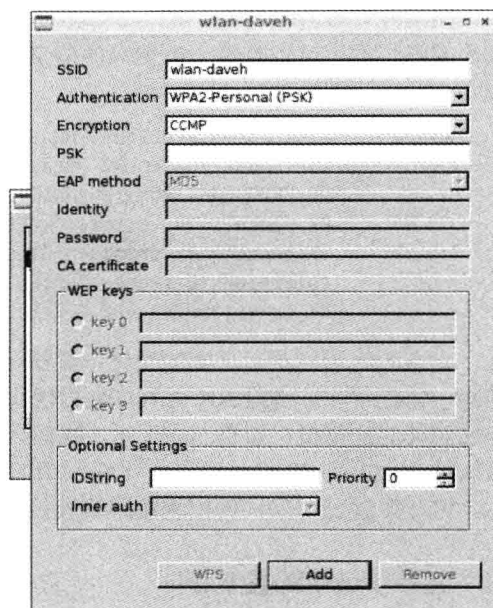


图 9-8 网络配置

- 在加密方式（Encryption）中选择加密方式。
- 在 PSK（preshared key）中填写你的无线网络密码。

选择正确的验证方式、填写正确的密码后，单击添加（Add）按钮，树莓派将保存你的无线网络设置并尝试连接，完成后应该会看到如图 9-9 所示的界面。

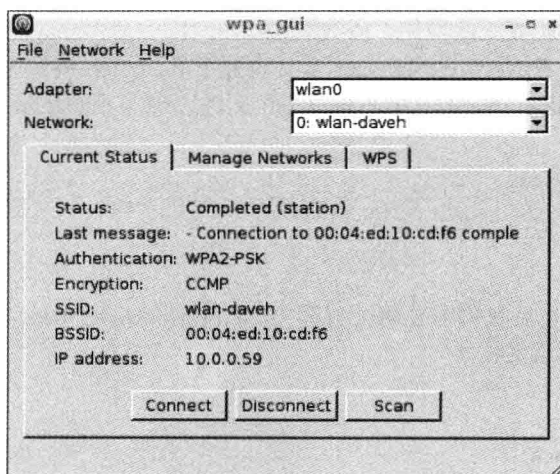


图 9-9 WiFi 连接成功

如果没有看到该界面，看一下弹出的错误消息，检查是否选择了正确的验证方式并填写了正确的密码，你还可以查看一下填写的信息是否和路由器中的配置相同，或者还可以使用 `ifconfig` 命令查看无线网卡的配置。

```
wlan0    Link encap:Ethernet  HWaddr 90:94:e4:51:81:7a
          inet addr:10.0.0.59  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:772 errors:0 dropped:10 overruns:0 frame:0
          TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:100749 (98.3 KiB)  TX bytes:4254 (4.1 KiB)
```

我们的例子显示了正确的配置，无线网卡得到了 IP 地址、网络掩码等，数据包的接收和发送也一切正常（分别是 RX 行和 TX 行现实的内容）。我们还可以用 `ping` 命令测试一下是否可以连通到其他计算机，如果一切正常，我们可以拔掉网线，并重启。完成后再使用其他计算机 `ping` 该 `wlan0` 设备的 IP 地址，看看无线网络的配置是否自动启动并已经连接上无线网络。测试成功后，树莓派就已经默认使用无线网络连接网络了。

在命令行中配置 WiFi

如果你不想使用图形化界面或者你想保持 Linux 的传统，完全通过命令行来管理系统，那也可以，这节我们介绍如何使用命令行管理无线网络。首先我们需要调整一下 `NetworkManager` 的配置，`NetworkManager` 根据以下的几个配置来决定是否管理指定的网络设备。

- 首先检查 `/etc/NetworkManager/NetworkManager.conf` 文件中是否打开了 `managed` 标志。
- 如果配置了 `managed` 标志，`NetworkManager` 将按照 `/etc/network/interfaces` 文件的配置管理网络设备。

如果你想通过命令行配置无线网络，首先需要禁止 `NetworkManager` 接管配置，如果你之前使用过图形化界面配置或者 `NetworkManager` 本身内置于 Raspbian 镜像中，那么你需要手动关掉它。打开 `/etc/NetworkManager/NetworkManager.conf` 文件，确保 `managed` 标志已经设置为 `false`。

```
[main]
plugins=ifupdown,keyfile

[ifupdown]
managed=false
```

完成后重启树莓派，之后我们需要将配置写入 `/etc/network/interfaces`，以便树莓派知道如何连接无线网络，让我们打开该文件。

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp
```

除了这些，如果你的系统已经自动配置了无线网卡，还会看到类似以下的内容。

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

配置中包括了本地回环 (lo) 设备和 eth0 设备的配置，本地回环的配置是自动管理 (auto) 状态，而 eth0 设备则设置为通过 DHCP 获得 IP 地址。如果 wlan0 设备已有一些自动配置内容，那么请先通过 # 注释掉这些内容或者直接删掉这些内容，我们通过下面的配置重新配置 wlan0 设备。

```
auto wlan0
iface wlan0 inet dhcp
    wpa-ssid <Your WiFi SSID>
    wpa-psk <Your WiFi password>
```

通过该节配置，Linux 系统将自动管理 wlan0 设备，通过 DHCP 获得 IP 地址。我们还指定了 SSID 和 PSK 密码，这样的配置兼容 WPA 和 WPA2 验证方式。完成配置后，我们重启网络接口，看看配置是否正确。

```
$ sudo ifdown wlan0
$ sudo ifup wlan0
Internet Systems Consortium DHCP Client 4.2.2
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
```

```
Listening on LPF/wlan0/90:94:e4:51:81:7a
Sending on   LPF/wlan0/90:94:e4:51:81:7a
Sending on   Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 8
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 15
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 9
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPOFFER from 10.0.0.1
DHCPACK from 10.0.0.1
bound to 10.0.0.59 -- renewal in 444671 seconds.
```

该输出显示 wlan0 设备成功连接了我们的无线网络并且通过 DHCP 获得了 IP 地址。如果你不想使用 DHCP 而想静态配置 IP 地址，我们可以修改 iface 行，使用 static 替代 dhcp，并在下面的配置中添加 IP 地址、子网掩码和网关等信息，最后得到的配置如下：

```
auto wlan0
iface wlan0 inet static
    address 10.0.0.57
    netmask 255.255.255.0
    gateway 10.0.0.1
    wpa-ssid <Your WiFi SSID>
    wpa-psk <Your WiFi password>
```

这样我们就完成了对无线网络的配置，重启树莓派，看看无线网络是否能够自动连接。如果连接失败，可能需要再检查一下关于 wlan0 的设置，包括 auto wlan0，并检查 NetworkManager 配置文件中的 managed=false 是否正确。

如果无线网络还是无法连接，再次检查一下 SSID 和密码是否正确，可以在命令行中执行 `ifup wlan0`，看系统是否有什么错误输出，并针对出现的错误修改配置。或者你还可以尝试移除其他的 USB 设备，因为多个设备有可能造成无线网卡供电不足。

最后你还可以尝试使用 `iw` 命令帮助查看信息或者查找错误。

- 执行 `iwlist scan` 搜索并列出附近的无线网络。
- 执行 `iwconfig` 显示无线网卡的配置细节。

几乎无线的树莓派

我们完成了整个配置，可以让树莓派完全无线连接网络了。现在可以把其他的有线网卡都移除了，只剩电源线连接在树莓派上，这样看起来清爽多了（见图 9-10）。

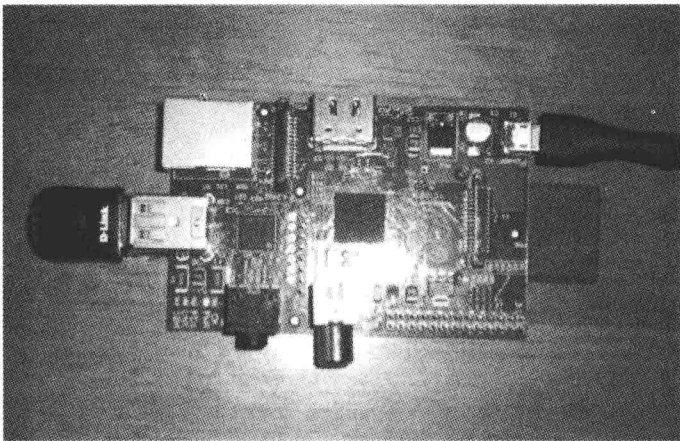


图 9-10 几乎无线的树莓派

9.4 移除电源线

想移除树莓派的电源线还是挺困难的，毕竟树莓派是个电子设备，它需要电。有限的几种方法中，最容易做到的就是使用一个 USB 外置电源。现在这种外置电源使用得很普遍，它可以为很多设备充电。直接将树莓派连接到 USB 外置电源，就完成了我们的目标，不过需要注意的是选择一个输出电流合适的外置电源，树莓派需要满足 1 安培的输出，有不少外置电源达不到这么高的输出。最后我们完成的无线树莓派看起来如图 9-11 所示。

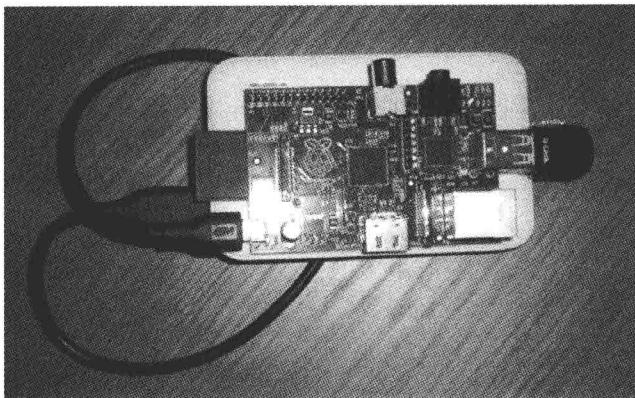


图 9-11 无线树莓派

当然你还可以进一步改造，使用充电电池或者太阳能电池为树莓派供电，我们这里就不再介绍了。

9.5 外置电源的 USB Hub

我们曾经提到过，如果在树莓派上连接过多的 USB 设备，树莓派可能无法正常工作，因为 USB 接口消耗了过多的电力。外置电源的 USB Hub 均通常使用一个 USB 接口连接到计算机上，并将其扩展成多个 USB 接口，另外它还通过电源线独立地为这些 USB 接口供电。这种 Hub 可以很好地解决 USB 接口供电问题，并且仍然能够保证设备的数据传输速度。如果你需要连接很多外置设备，那么这么一个外置电源 USB Hub 就很实用，它的样子如图 9-12 所示。

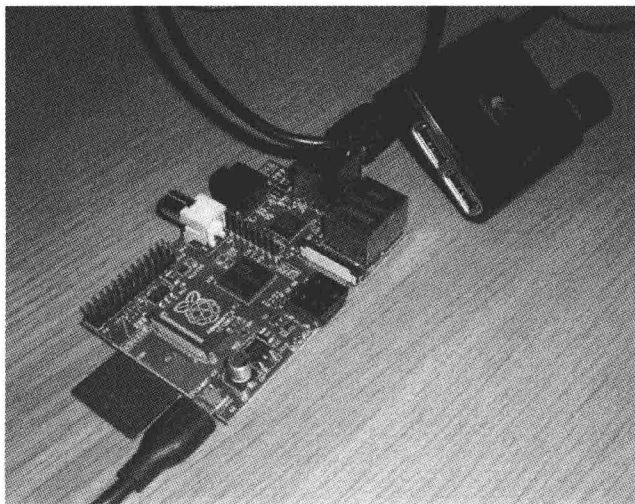


图 9-12 外置电源的 USB Hub

本章小结

本章中我们一起学习了如何“无线化”你的树莓派，我们首先为树莓派增加 VNC 连接能力，你不再只能使用 SSH 连接命令行控制台了，而是有了图形化界面。之后我们让树莓派连接上了无线网络，最后我们还设法移除了树莓派的电源线连接，让其成为真正的无线树莓派。

树莓监控派

大家都看过詹姆斯邦德，他总是有各种小玩意儿帮助他解决困境、拯救世界。树莓派可能不能拯救世界，不过它也可以帮助我们解决很多问题。树莓派只消耗很少的电力，可以安装不同的外设，因此我们可以用树莓派做到很多在 007 电影中见过的事情。

007 经常需要监控房间是否有人进入，并发出报警，我们可以用树莓派做同样的事情。将摄像头连接到树莓派，连接网络，并做一些配置，我们就可以监控房间的动静或者看看你的宠物白天在干什么。

10.1 需要的原料

本章我们需要的所有东西如下所示（见图 10-1）。

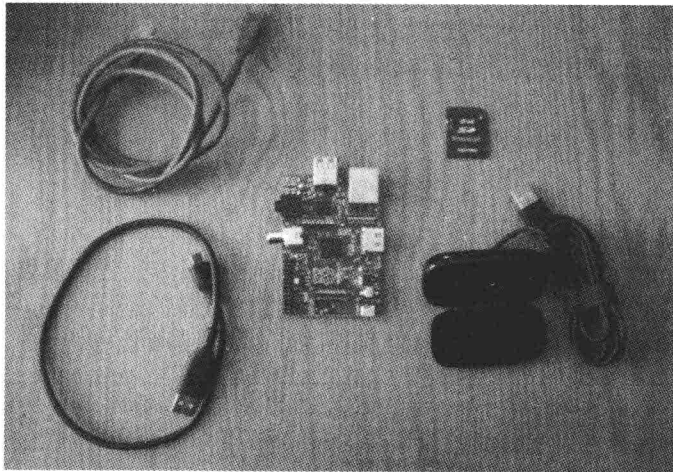


图 10-1 树莓监控派

- 树莓派×1。
- USB 电源×1。

- 网线 x1。
- USB 网络摄像头 x1。
- SD 卡 x1。

需要注意的是并非所有网络摄像头都兼容树莓派，本章我们将以罗技 C525 为例子介绍。不过现在树莓派已经能够兼容很多硬件，所有兼容的网络摄像头型号都可以在以下地址查询 http://elinux.org/RPi_VerifiedPeripherals#USB_Webcams，你可以检查一下你买的摄像头是否能够兼容树莓派。

10.2 准备

准备步骤很简单，接上外置电源，连好网线，插入 SD 卡，这里先不连接摄像头，需要做的就这么多。不过完成本章的内容还需要以下的知识。

- 安装和使用 Raspbian 系统。
- 了解如何使用 Raspbian 系统的命令行环境。

如果你还不熟悉以上两点，没关系，可以先阅读本书之前的章节，学习相关的知识，之后再来阅读本章。

10.3 开始动手

让我们启动树莓派，设置好网络连接。如果你的树莓派还没有安装好系统，那么可以参考之前的章节准备好 Raspbian 系统，并启用 SSH 远程连接，我们将使用远程连接配置并管理树莓监控派。

通过 SSH 远程连接登录树莓派后，先执行一下 `dmesg` 命令，看看有什么输出。

dmesg 命令

`dmesg` 是 `driver message` 的缩写，用于输出所有的内核日志，内核日志也包括了系统中的设备驱动日志。该命令在管理外设时很有用，可以用来了解设备的运行状态，该命令还可以用于调试内核和硬件驱动中的问题。让我们来看看执行该命令会得到什么输出。

注意 Linux 用户信奉大道至简 (less is more)，当你执行某条命令后却发现输出的内容超出了屏幕可以显示的长度，你可以重新执行该命令，并且加上管道将输出导入 `less` (例如 `dmesg | less`)，这样你就可以使用上下箭头和空格键在输出内容中翻页。

```
[ 1.998581] Waiting for root device /dev/mmcblk0p2...
[ 2.071268] mmc0: new high speed SD card at address 7d37
[ 2.079917] mmcblk0: mmc0:7d37 SD02G 1.83 GiB
```

```

[ 2.088723] mmcblk0: p1 p2
[ 2.132278] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[ 2.145275] VFS: Mounted root (ext4 filesystem) on device 179:2.
[ 2.155272] Freeing init memory: 200K
[ 2.187333] usb 1-1: new high speed USB device number 2 using dwc_otg
[ 2.418045] usb 1-1: New USB device found, idVendor=0424, idProduct=9512
[ 2.427750] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 2.438708] hub 1-1:1.0: USB hub found
[ 2.445041] hub 1-1:1.0: 3 ports detected
[ 2.727674] usb 1-1.1: new high speed USB device number 3 using dwc_otg
[ 2.837920] usb 1-1.1: New USB device found, idVendor=0424, idProduct=ec00
[ 2.857413] usb 1-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 2.878968] smsc95xx v1.0.4
[ 2.942985] smsc95xx 1-1.1:1.0: eth0: register 'smc95xx' at usb-bcm2708_usb-1.1, smc95xx
USB 2.0 Ethernet, b8:27:eb:8a:46:ba
[ 11.006171] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[ 11.426960] ### snd_bcm2835_alisa_probe c05c88e0 ##### PROBING FOR bcm2835 ALSA
device (0):(1) #####
[ 11.442869] Creating card...
[ 11.448268] Creating device/chip ..
[ 11.454771] Adding controls ..
[ 11.460340] Registering card ....
[ 11.475463] bcm2835 ALSA CARD CREATED!
[ 11.487786] ### BCM2835 ALSA driver init OK ###
[ 18.493739] smsc95xx 1-1.1:1.0: eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[ 24.672322] Adding 102396k swap on /var/swap. Priority:-1 extents:1 across:102396k SS

```

上面就是 `dmesg` 的输出例子，似乎比较难以理解。让我们先了解一下 `dmesg` 的输出格式。每条日志的开头都以 `[]` 块开头，其中的数字表示的是系统启动到日志输出经过的时间，可以看到日志是按照时间顺序输出的。

日志的主要内容大多用于诊断内核和驱动的运行状态，每条日志都包含了大量的信息，例如第一条日志：

```
waiting on root device /dev/mmcblk0p2
```

这一行的内容显示系统正在等待设备 `/dev/mmcblk0p2`，不过这是什么设备呢？开始的 `/dev` 我们已经介绍过，是系统的设备虚拟文件目录，Linux 系统中的所有设备都以虚拟文件的形式存放在 `/dev` 目录中，`mmcblk0p2` 这个名称的含义我们这样理解：

- `mmc` 表示多媒体存储卡（multimediacard，一种闪存标准）。
- `blk` 表示块设备。
- `0` 表示该设备上的第一个逻辑块。
- `p2` 表示该块上的第二个分区。

可以看到系统正在等待 SD 卡，从第 2 行到第 4 行显示了 SD 卡顺序注册了其中的每一个子设备，之后系统所等待的分区被挂载到系统。完成了 SD 卡的注册和挂载后，树莓派又接着注册了 USB 设备（8~15 行）、网卡（16、17 和 26 行）和 Advanced Linux Sound Architecture（ALSA）

高级音频设备（19~25 行）。

之所以我们没有在启动前连接 USB 摄像头是因为我们想通过 `dmesg` 检查一下连接后会发生什么。现在让我们将摄像头连接到树莓派，再次执行 `dmesg` 看看有什么新的输出。

```
[ 8168.793423] usb 1-1.2: new high speed USB device number 4 using dwc_otg
[ 8169.147691] usb 1-1.2: New USB device found, idVendor=046d, idProduct=081d
[ 8169.147736] usb 1-1.2: New USB device strings: Mfr=0, Product=0, SerialNumber=1
[ 8169.147758] usb 1-1.2: SerialNumber: 8627F4C0
[ 8169.314171] Linux video capture interface: v2.00
[ 8169.336482] uvcvideo: Found UVC 1.00 device <unnamed> (046d:081d)
[ 8169.402071] input: UVC Camera (046d:081d) as /devices/platform/bcm2708_usb/usb1/1-1/1-1.2/1-1.2:1.2/input/input0
[ 8169.402280] usbcore: registered new interface driver uvcvideo
[ 8169.402299] USB Video Class driver (1.1.1)
[ 8169.460903] usbcore: registered new interface driver snd-usb-audio
```

这几行 `dmesg` 输出和之前的日志内容有明显的不同，主要是时间明显要大于之前的日志时间。前 4 行日志显示 USB 接口发现了新插入的设备，并将其识别为 `dwc_otg` (DesignWare Cores-On The Go 的缩写) 设备。第 5 行显示发现一个视频捕捉设备，第 6 行显示 `uvcvideo` 模块发现并注册了一个 USB Video Class (UVC) 设备，最后一行显示注册了摄像头的麦克风，这些输出表明摄像头已经正常地工作了。

摄像头的安装已经完成，不过我们还需要找到该设备的虚拟文件路径，以便让应用程序能够访问。`dmesg` 的日志输出中并没有写出设备的路径是什么，所以我们还是查看一下 `/dev` 目录中有什么（见图 10-2）。

```
$ ls /dev/
```

```
pi@raspberrypi ~ $ ls /dev/
autofs          loop0           network_latency ram4            tty0            tty23           tty38           tty52           ttyAMA0         vcs5
block           loop1           network_throughput ram5            tty1            tty24           tty39           tty53           ttyprintk       vcs6
btrfs-control   loop2           null            ram6            tty10           tty25           tty4            tty54           uinput          vcsa
bus             loop3           ppp             ram7            tty11           tty26           tty40           tty55           urandom          vcsa1
cachefiles      loop4           pts            ram8            tty12           tty27           tty41           tty56           usbdev1.1        vcsa2
char            loop5           ram0            ram9            tty13           tty28           tty42           tty57           usbdev1.2        vcsa3
console         loop6           ram1            random          tty14           tty29           tty43           tty58           usbdev1.3        vcsa4
cpu_dma_latency loop7           ram10           raw            tty15           tty3            tty44           tty59           usbdev1.4        vcsa5
disk            loop-control    ram11           root            tty16           tty30           tty45           tty6            v4l              vcsa6
fb0             MAKEDEV         ram12           shm            tty17           tty31           tty46           tty60           vchiq            video0
fd             mapper          ram13           snd            tty18           tty32           tty47           tty61           vc-mem           xconsole
full           mem            ram14           sndstat        tty19           tty33           tty48           tty62           vcs              zero
fuse           mmcblk0         ram15           stderr         tty2            tty34           tty49           tty63           vcs1
input          mmcblk0p1       ram16           stdin          tty20           tty35           tty5            tty7            vcs2
kmsg           mmcblk0p2       ram2            stdout         tty21           tty36           tty50           tty8            vcs3
lpc            net            ram3            tty            tty22           tty37           tty51           tty9            vcs4
```

图 10-2 /dev 目录中的文件

看起来有很多的设备，让我们想办法过滤掉一些，包括 `tty` 基本输入输出设备、`ram` 内存设备、`loop` 本地回环连接设备、`memblk` SD 卡设备和 `vcs` 虚拟控制台设备。这些都不可能是我们寻找的 USB 摄像头，不过似乎还是剩下很多设备。让我们直接一些，我们寻找的是视频相关的设备，因为 USB 摄像头被注册为了视频捕捉设备，所以应该就是 `video0` 了。不过我们怎么能够确认呢？

我们可以使用 `udev` 管理命令，`udev` 是 Linux 内核的设备管理模块，它控制设备如何注册到计算机上，管理应用程序如何访问设备。在 Debian（树莓派的 Raspbian）系统中，使用 `udevadm` 命令调用该模块。

```
$ udevadm info -q all -n /dev/video0
P: /devices/platform/bcm2708_usb/usb1/1-1/1-1.2/1-1.2:1.2/video4linux/video0
N: video0
S: v4l/by-id/usb-046d_081d_8627F4C0-video-index0
S: v4l/by-path/platform-bcm2708_usb-usb-0:1.2:1.2-video-index0
E: DEVLINKS=/dev/v4l/by-id/usb-046d_081d_8627F4C0-video-index0 /dev/v4l/by-path/platform-bcm2708_usb-usb-0:1.2:1.2-video-index0
E: DEVNAME=/dev/video0
E: DEVPATH=/devices/platform/bcm2708_usb/usb1/1-1/1-1.2/1-1.2:1.2/video4linux/video0
E: ID_BUS=usb
E: ID_MODEL=081d
E: ID_MODEL_ENC=081d
E: ID_MODEL_ID=081d
E: ID_PATH=platform-bcm2708_usb-usb-0:1.2:1.2
E: ID_PATH_TAG=platform-bcm2708_usb-usb-0_1_2_1_2
E: ID_REVISION=0010
E: ID_SERIAL=046d_081d_8627F4C0
E: ID_SERIAL_SHORT=8627F4C0
E: ID_TYPE=video
E: ID_USB_DRIVER=uvcvideo
E: ID_USB_INTERFACES=:010100:010200:0e0100:0e0200:
E: ID_USB_INTERFACE_NUM=02
E: ID_V4L_CAPABILITIES=:capture:
E: ID_V4L_PRODUCT=UVC Camera (046d:081d)
E: ID_V4L_VERSION=2
E: ID_VENDOR=046d
E: ID_VENDOR_ENC=046d
E: ID_VENDOR_ID=046d
E: MAJOR=81
E: MINOR=0
E: SUBSYSTEM=video4linux
E: TAGS=:udev-acl:
E: UDEV_LOG=3
E: USEC_INITIALIZED=8168852755
```

该命令给了我们一大堆输出，这些都是与 `/dev/video0` 设备相关的信息，我们不需要看完所有的输出，前几行就已经足够了。在第一行中我们看到了摄像头在 `dmesg` 日志中的设备路径，所以 `/dev/video0` 就是我们刚刚接入的 USB 网络摄像头。

排错

如果从 `dmesg` 的日志中没有看到你的摄像头注册到系统，那么可能需要手动安装摄像头的驱动程序，你可以从生产商的网站查找对应的驱动。如果驱动安装正确，并且 `dmesg` 日志显示视频

捕捉设备也成功注册，那么可以按照上面的方法查找你的摄像头对应的设备文件路径。

如果摄像头安装正确并成功注册，系统中却不存在/dev/video0 设备，你也许需要仔细查看其他的设备名称或者仔细查看 dmesg 日志。虽然 Linux 尽量保证/dev 中的设备有合理的名称，但是厂商也有可能按照自己的意愿使用别的名称。所以如果按照我们上面的方法没有找到你的摄像头设备路径，请仔细查看所有的信息。

10.4 运动捕捉

上一节，我们完成了网络摄像头的安装，这一节我们将使用摄像头来做一些事情。首先我们需要在树莓派上安装 motion 运动捕捉应用程序。

```
$ sudo apt-get install motion
```

该命令的输出如下：

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  ffmpeg libav-tools libavcodec53 libavdevice53 libavfilter2 libavformat53 libavutil51
  libdc1394-22 libdirac-You
  libjack-jackd2-0 libmp3lame0 libpostproc52 libpq5 libraw1394-11 libschrödinger-1.0-0
  libspeex1 libswscale2 libtheora0 libva1
  libvpx1 libx264-123 libxvidcore4
Suggested packages:
  jackd2 libraw1394-doc speex mysql-client postgresql-client
The following NEW packages will be installed:
  ffmpeg libav-tools libavcodec53 libavdevice53 libavfilter2 libavformat53 libavutil51
  libdc1394-22 libdirac-encoder0 libgsm1
  libjack-jackd2-0 libmp3lame0 libpostproc52 libpq5 libraw1394-11 libschrödinger-1.0-0
  libspeex1 libswscale2 libtheora0 libva1
  libvpx1 libx264-123 libxvidcore4 motion
0 upgraded, 24 newly installed, 0 to remove and 71 not upgraded.
Need to get 8,365 kB of archives.
After this operation, 17.6 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

这样就安装完成了，我们使用以下命令运行 motion。

```
$ motion -s
```

该命令运行后，我们的摄像头开始工作了，在摄像头前做些动作，会看到命令行有了不同的输出。

```
[1] Changes: 3374 - noise level: 15
[1] Changes: 3198 - noise level: 15
[1] Changes: 3011 - noise level: 15
```

```

[1] Changes: 2922 - noise level: 15
[1] Changes: 2555 - noise level: 15
[1] Changes: 2390 - noise level: 15
[1] Changes: 2491 - noise level: 15
[1] Changes: 2874 - noise level: 15
[1] Changes: 2817 - noise level: 15
[1] Changes: 3238 - noise level: 15
[1] Changes: 3093 - noise level: 15
[1] Motion detected - starting event 1
[1] File of type 1 saved to: ./01-20120910203217-05.jpg
[1] Changes: 2912 - noise level: 15
[1] File of type 1 saved to: ./01-20120910203217-06.jpg
[1] Changes: 2480 - noise level: 15
[1] File of type 1 saved to: ./01-20120910203217-07.jpg

```

输出的信息显示 motion 捕捉了我们的动作，当动作幅度大于指定的 noise 值时，它将捕捉图像并保存至文件，如果你的 SD 卡读写速度比较慢，可能需要一些时间完成文件输出。如果你想停止 motion，可以使用 Ctrl+C 组合键。我们可以检查一下刚才捕捉到的图片，通过查看图片，可以了解摄像头是否工作正常或者需要调校。由于我们使用的是 SSH 登录树莓派，所以查看图片似乎不太容易。我们可以使用 scp 系列的工具，将图片远程拷贝到其他计算机中查看。在 Windows 系统中，可以使用 WinSCP 拷贝文件，在 Linux 或者 Mac 系统中，则可以直接使用 scp 命令。

```
$ scp <raspberrypi's ip>:/home/pi/*.jpg.
```

 **注意** 注意该句最后的点 (.) 表示的是拷贝目的地，也就是当前目录。

该命令通过 IP 地址访问树莓派，将/home/pi 目录中的所有以.jpg 结尾的文件拷贝出来，最后的点 (.) 表示拷贝的目标目录，在 Linux 系统中点 (.) 表示当前目录，也就是将树莓派中刚才捕捉到的运动图片拷贝到当前目录。

我们已经能够拷贝文件了，不过作为监控器，我们需要让 motion 能够自动启动，而不是由我们手动控制，所以还需要做一些设置。

 **注意** 本章接下来的所有设置都需要使用 root 权限，所以启动文本编辑器时请使用 sudo 提高权限。

首先，打开/etc/default/motion 文件，找到 start_motion_daemon=no 这一行，将其修改为 start_motion_daemon=yes，该行表示在使用 start 命令后，motion 将自动启动。之后，我们调整 motion 的配置文件，打开/etc/motion/motion.conf 文件，将守护模式 (daemon mode) 从 off 修改为 on，该配置将使 motion 以后台守护进程的方式运行。我们还需要调整 videodevice 配置，填写之前找到的设备路径（例如我们摄像头对应的/dev/video0）。

基本的启动配置调整完毕后，还可以根据需要调整摄像头捕捉画面的高度 (height)、宽度 (width) 和帧率 (framerate)。默认配置是每秒捕捉两张图片，图片大小为 320 像素 x240 像素，

文件大小大约为 12KB。你可以根据自己的需要调整这些设置，不过请注意图片大小不要超过摄像头最高分辨率。

另一个重要的配置是 `threshold`，当 `motion` 发现图像的变化超过该值时，将会捕捉画面。目前我们不建议调整默认值，如果你需要动作捕捉更加敏感，可以减小该值。

还可以调整 `ffmpeg_cap_new` 配置项，该配置表示是否需要捕捉运动视频。我们将其设置为 `off`，你也可以打开该功能。`motion` 默认将视频编码为 `.swf` 文件，你可以通过调整 `ffmpeg_video_codec` 修改编码格式。

最后一个可能需要调整的配置项是 `target_dir`，该配置项可以调整运动捕捉图片的存储目录，默认使用 `/tmp/motion` 目录，该目录位于 `/tmp` 之中，所以会在每次重启后清空。有时自动清理机制还是挺方便的，不过如果你需要保存这些文件请及时拷贝或者将该配置路径调整到其他目录。

保存配置文件，我们就完成了所有设置，下面是我们树莓派上的配置文件完整内容。

```
daemon on
process_id_file /var/run/motion/motion.pid
setup_mode off
videodevice /dev/video0
v4l2_palette 8
input 8
norm 0
frequency 0
rotate 0
width 320
height 240
framerate 2
minimum_frame_time 0
netcam_tolerant_check off
auto_brightness off
brightness 0
contrast 0
saturation 0
hue 0
roundrobin_frames 1
roundrobin_skip 1
switchfilter off
threshold 1500
threshold_tune off
noise_level 32
noise_tune on
despeckle FedDL
smart_mask_speed 0
lightswitch 0
minimum_motion_frames 1
pre_capture 0
post_capture 0
gap 60
max_mpeg_time 0
```

```

output_all off
output_normal on
output_motion off
quality 75
ppm off
ffmpeg_cap_new off
ffmpeg_cap_motion off
ffmpeg_timelapse 0
ffmpeg_timelapse_mode daily
ffmpeg_bps 500000
ffmpeg_variable_bitrate 0
ffmpeg_video_codec swf
ffmpeg_deinterlace off
snapshot_interval 0
locate off
text_right %Y-%m-%d\n%T-%q
text_changes off
text_event %Y%m%d%H%M%S
target_dir /tmp/motion
snapshot_filename %v-%Y%m%d%H%M%S-snapshot
jpeg_filename %v-%Y%m%d%H%M%S-%q
movie_filename %v-%Y%m%d%H%M%S
timelapse_filename %Y%m%d-timelapse
webcam_port 8081
webcam_quality 50
webcam_motion off
webcam_maxrate 1
webcam_localhost on
webcam_limit 0
control_port 8080
control_localhost on
control_html_output on
track_type 0
track_auto off
track_motorx 0
track_motory 0
track_maxx 0
track_maxy 0
track_iomomo_id 0
track_step_angle_x 10
track_step_angle_y 10
track_move_wait 10
track_speed 255
track_stepsize 40

```

完成配置，我们准备来测试一下，我们将使用 `start` 命令启动 `motion` 的守护进程，该命令也会在树莓派启动时被自动调用，所以如果我们手动启动成功，以后每次重启树莓派后，`motion` 也将会自动启动。执行 `sudo /etc/init.d/motion start`，我们看到摄像头上的灯亮了。之后我们在摄

像头前做一些动作，应该会看到/tmp/motion（或者你设置的其他 target_dir 路径）中出现了运动捕捉图片。我们还可以通过以下命令查看 motion 是否已在后台启动。

```
$ ps -ef | grep motion
```

排错

如果你的摄像头或 motion 不能正常工作，那么最好重新按照本章之前的内容，拔下并再次插入摄像头，检查 dmesg 的日志输出，使用 udev 查找设备路径。还可以使用 apt-get 命令重新安装 motion。

```
$ apt-get --reinstall install motion
```

使用 motion -s 命令启动 motion，检查是否运行正常，如果有错误，检查错误消息内容。

如果仍然不能正常工作，请再次检查配置文件/etc/default/motion 的内容。如果已经运行却无法得到运动捕捉文件，请检查/var/log/messages 内容，看看是否有与 motion 相关的错误消息。

10.5 自动报警

到这里我们完成了网络摄像头的连接，也完成了监控程序的安装和配置，树莓监控派还差一个功能，当系统捕捉到运动图像后，我们希望它能够发出通知。

我们将使用一种常见且简单的通信方式：e-mail。我们不是要自己搭建一个邮件服务器，这样做太复杂了。我们将安装一个轻量级的邮件客户端，并且将通知以邮件的形式发送给指定的地址，这样当有动作被捕捉到时，我们就可以通过邮件来查看发生了什么。

安装 SSMTP

我们在树莓监控派中使用 SSMTP 应用程序来发送通知邮件，SSMTP 是一个轻量级的邮件传输代理程序（mail transfer agent, MTA），系统管理员常常使用这个程序从服务器上发送系统报警通知。SSMTP 最大的优点在于配置简单，所以很适合在树莓派上使用。

我们使用以下命令来安装 SSMTP。

```
$ sudo apt-get install ssmtp
```

执行后，你会看到类似下面的结果。

```
$ sudo apt-get install ssmtp
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libgnutls-openssl27
The following NEW packages will be installed:
  libgnutls-openssl27 ssmtp
0 upgraded, 2 newly installed, 0 to remove and 71 not upgraded.
```

```
Need to get 272 kB of archives.
After this operation, 279 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

安装完成后，还需要做一些配置。SSMTP 的配置文件路径是/etc/ssmtp/ssmtp.conf，我们使用文本编辑器打开并修改其中的配置。

首先配置的是邮件的发送地址，你可能想用不同的邮箱来发送通知邮件，本节我们将以 Gmail 为例子进行配置，如果你想使用其他的邮件服务商，可以相应地修改这些配置。

发送地址对应的重要配置是 mailhub，SSMTP 将邮件传送到该配置指定的服务器，常见的地址格式是 mail.<yourdomain>.<whatever>，不过 Gmail 的 SMTP 服务器地址有所不同，是 smtp.gmail.com:587。

你会发现，我们在服务器地址后面加上了:587，这表示指定连接 587 端口而不是 SMTP 默认的 25 端口。该端口表明 Gmail 使用的是 Transport Layer Security (TLS) 加密 e-mail 传输协议，由于使用了 TLS，所以我们需要调整以下两项内容。

```
UseTLS=YES
UseSTARTTLS=YES
```

接下来我们配置 hostname，该项用来表示 e-mail 发自树莓监控派，如果你拥有域名，填写你的域名，如果没有的话，可以填写例如 raspberry.spi 之类的名字。同时确保下一项 FromLineOverride=yes，该项使我们可以修改 e-mail 的“from”字段。

最后，我们配置用户验证，由于使用 Gmail 发送邮件，我们需要提供自己的账号以便通过 Google 的用户验证。我们主要配置 3 个项目，首要的两个就是用户名和密码，也就是配置文件中的 AuthUser=username@gmail.com 和 AuthPass=password。第 3 项配置是 AuthMethod，用于确定 SMTP 服务器的账号验证方式。Google 服务器使用 LOGIN 验证方法，所以我们设置 AuthMethod=LOGIN。

这样我们的配置就完成了，完整的配置文件例子如下：

```
root=
Mailhub=smtp.gmail.com:587
UseTLS=YES
UseSTARTTLS=YES
Hostname=raspberrypi
FromLineOverride=yes
AuthUser=username@gmail.com
AuthPass=password
AuthMethod=LOGIN
```

 **注意** 该文件中包含重要的账户和密码，我们将限制读写权限为 root 用户以确保其他人无法看到文件内容，不过请设置一个较强的 root 密码。

最后我们设置该文件的访问权限，以确保只有有限用户能够读写该文件的内容，防止其他人看到我们的 Gmail 账号和密码，我们执行以下命令。

```
$ sudo chmod 640 /etc/ssmtp/ssmtp.conf
```

该命令设置文件仅仅可以由 root 用户操作，由 mail 组用户读取。

完成了 SSMTP 的配置后，我们还需要做一件事，就是将运动捕捉和邮件发送联系起来。当 motion 捕捉到运动图片后，调用 SSMTP 发送邮件通知我们，帮助我们完成这项工作的工具是 mutt。

安装配置 mutt

mutt 是一个命令行 e-mail 客户端，它的开发者说：“所有的邮件客户端都很糟，我只是没那么糟”。mutt 可以发送带有文件附件的邮件，这也正是我们想要的。

我们先通过以下命令安装 mutt。

```
$ sudo apt-get install mutt
```

执行后，你会看到类似下面的结果。

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libpgpme11 libpth20 libtokyocabinet9
Suggested packages:
  gpgsm gnupg2 urlview mixmaster
The following NEW packages will be installed:
  libpgpme11 libpth20 libtokyocabinet9 mutt
0 upgraded, 4 newly installed, 0 to remove and 71 not upgraded.
Need to get 1,985 kB of archives.
After this operation, 7,181 kB of additional disk space will be used.
```

这样我们就完成了 mutt 的安装，mutt 是一个轻量级的 e-mail 客户端，所以配置也很简单，我们只需要提供 MTA 程序的路径（也就是我们安装的 SSMTP 的路径）即可。配置方法是在树莓派的用户主目录增加一个 .muttrc 文件，该文件仅需要一行内容。

```
set sendmail="/usr/sbin/ssmtp"
```

我们不准创建文件再打开编辑内容，而是使用以下的一条命令完成整个过程。

```
$ echo "set sendmail=\"/usr/sbin/ssmtp\" " > ~/.muttrc
```

这样就完成了。

注意 引号中的引号需要使用反斜杠 (\) 来转义。

检查一下 root 用户的主目录中的 .muttrc 文件¹，这就是我们刚才的配置。mutt 将会从用户主目录中读取 .muttrc 的内容，为我们调用 SSMTP 发送邮件。很多应用程序都是用 rc 文件存储配置的，所以如果你找不到某个应用程序的配置文件，可以尝试找找以 rc 结尾的文件。

¹ 需要以 root 权限执行之前的命令。

测试报警系统

完成了 SSMTP 和 mutt 的安装与配置，让我们来测试一下。我们将使用 mutt 命令向我们自己的邮箱发送邮件。

```
$ echo "the quick brown fox jumps over the lazy dog" | sudo mutt -s "[INTRUDER ALERT]
Test of intruder system" <your email>@gmail.com
```

后面我们将会解释命令中的每个部分，不过先来看一下发送的邮件内容，查看我们的收件箱，会看到如图 10-3 所示的内容。

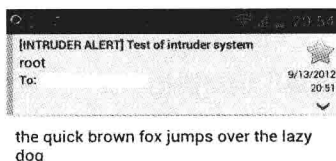


图 10-3 树莓派发来的邮件

刚才的命令可以被分解为两个部分，每个部分都有固定的语法。

```
echo <mail content> | mutt -s <subject> <recipient>
```

因为 mutt 通过输入而不是命令行参数接收邮件的内容，所以这里我们使用了 echo 命令向 mutt 输入邮件内容。

注意 邮件的内容需要使用双引号包围起来，以确保不会被错误地理解为命令行参数，双引号中的引号需要使用反斜杠转义。

之所以使用 mutt，是因为它可以发送文件附件，可以将树莓监控派的动作捕捉图片通过邮件发送给我们。我们需要调整一下之前的命令格式。

```
echo <mail content> | mutt -s <subject> -a <filename> -- <recipient>
```

该命令中 -a 参数开关指定了文件附件的路径，-- 则用于区分文件路径参数和接收邮箱参数。

排错

如果配置中出现了问题，你可以尝试重新配置 SSMTP 和 mutt。注意发送邮件时是否出现了登录错误，检查一下用户名密码是否配置正确。是不是出现了不支持的验证方式错误，检查一下 TLS 设置和 AuthMethod 设置。检查是否正确地设置了邮件服务器地址，检查是否有多余的空格，或者是不是权限配置错误，导致无法读取配置文件。

如果你尝试使用 pi 用户调用 mutt 发送邮件，可能会出现错误，因为 pi 用户并非 mail 用户组的成员，所以无法读取我们的配置。如果你确实想使用 pi 用户，那么通过 gpasswd 命令将 pi 用户加入 mail 用户组。

```
$ gpasswd -a <username> mail
```

到这里，我们在树莓监控派上完成了运动捕捉配置，也拥有了发送报警邮件的能力，下面一节，我们将让整个过程自动化。

10.6 模块拼接

现在我们已经完成了如下工作。

- 为树莓派添加摄像头。
- 安装配置 motion 运动捕捉应用程序。
- 通过摄像头和程序的配合来捕捉运动图像。
- 安装配置邮件传输代理程序 SSMTP。
- 安装配置邮件客户端 mutt。
- 通过 SSMTP 和 mutt 实现在命令行下发送邮件。

以上所有的操作我们都是手工完成的，那么如何使这些操作组合起来，并实现自动化呢？我们需要实现这样的功能：摄像头捕捉到运动图像，mutt 将图像以邮件的形式发给我们。虽然肯定有人想去找现成的解决方案，但实际上我们完全可以自己实现。

需求分析

软件设计的第一要务就是充分理解需求，在这里，我们的需求如下：

- 以摄像头捕捉到运动图像作为后续操作的触发条件。
- 发送一封包含该运动图像的邮件。
- 程序应该可以自动执行。

以上就是我们需要完成的主要需求。此外还有一些注意事项，罗列如下：

- 只发送最新图像。
- 只发送由 motion 捕捉到的图像。

- 程序应该可以一直运行在后台。

下一步我们需要关心的就是一些细节问题了。

- 邮件中应该附上捕捉到图像的时间。
- 应该频繁检查和发送图片，实现准实时监控。

技术选型

在实际动手实现前，我们需要先根据需求决定要使用什么工具和编程语言。在本例中我们需要用到不少命令，列出所需的图片并通过命令行进行邮件发送。所以使用在前面章节中介绍过的 BASH 应该是个不错的选择。按启动方法来分，实现方案大致有两类。

- 第一种，让程序持续运行，并通过循环不断检查是否捕捉到了新图像。
- 第二种，使用 `cron` 命令定期执行一次程序。

处理图像的方法也有两种。

- 第一种，删除已发送的图像，避免空间浪费。
- 第二种，保留所有图像，每次只发送新捕捉的图像。

这样我们就有了 4 种可能的实现方案。选择往往是困难的，让我们降低一下难度，下面是较为合理的两种方案。

- 通过 `cron` 定期执行，搜索图像并发送，完成后删除图像。
- 程序持续运行，监控图像文件变动，并发送最新的图像。

设计解决方案

现在我们有两种可行的解决方案，让我们先来看看它们有什么共同点，即都需要获取 `motion` 存放图像的文件夹路径。我们当然可以将文件夹路径直接保存在一个变量里，但这显然不是 Linux Style。正确的做法是从 `motion` 配置文件中读取路径，这里我们就需要引入一个变量来保存这一配置文件的路径，即 `MOTIONCONF=/etc/motion/motion.conf`。

 **注意** 下文中的操作都要以 `root` 用户权限执行（别忘了我们可以通过 `sudo` 或 `su` 命令获得这一权限）。

接下来我们要取出 `motion` 存放图像的文件夹路径。使用 `grep` 命令就可以轻易地取出特定行，这里要找的是以 “`target_dir`” 开头（用 `grep` 的语法表示为：“`^target_dir`”）的那一行。

```
root @raspberrypi ~ $ grep "^target_dir" $MOTIONCONF
target_dir /tmp/motion
```

我们需要的是 “`/tmp/motion`” 这部分内容。`awk` 命令默认以空格来分隔数据，通过它我们就可以取出我们所需的数据。

```
root@raspberrypi ~ $ grep "^target_dir" $MOTIONCONF | awk '{ print $2}'
/tmp/motion
```

接下来我们要把结果保存到一个变量中。

```
MOTIONDIR=' grep "^target_dir" $MOTIONCONF | awk '{ print $2}''
```

如果 `motion` 暂时还没有捕捉到图像，这个文件夹很可能还没有被创建。安全起见，我们可以先做一个检查，如果文件夹还不存在，就创建这个文件夹。通过 `BASH` 检查一个文件夹存在与否的语法是这样的：`if [ -d <directory name> ]`。这里我们需要在文件夹不存在时进行操作，所以在“-d”前面还要加上逻辑非的“!”。

```
if [ ! -d $MOTIONDIR ]; then
    mkdir $MOTIONDIR
fi
```

创建文件夹后我们需要将文件夹的所有者从 `root` 用户改为 `motion` 用户。

```
if [ ! -d $MOTIONDIR ]; then
    mkdir $MOTIONDIR
    chown motion $MOTIONDIR
fi
```

文件夹准备好后我们再来看如何实现文件夹的监控。这里我们需要使用 `find` 命令在文件夹中查找“.jpg”文件，需要用到两个参数。

- `-maxdepth 1` 表示不检查子文件夹。
- `-type f` 表示只查找文件而非文件夹。

```
LISTFILES=('find $MOTIONDIR -maxdepth 1 -type f')
```

保险起见，我们需要再通过 `grep` 命令来确保取到的都是“.jpg”文件。

```
LISTFILES=('find $MOTIONDIR -maxdepth 1 -type f | grep jpg$')
```

我们还可以加上 `wc -l` 命令来获得文件夹中的文件数。

```
NUMFILES=' find $MOTIONDIR -maxdepth 1 -type f | grep jpg$ | wc -l'
```

接下来我们要对原来使用 `mutt` 发送邮件的代码做些小修改，来添加图像附件，并通过“`mutt -F /root/.muttrc`”来制定我们的配置文件。

```
echo -e "Warning,\nAn intruder was detected at 'date'\nPlease see the image attached for details"|
mutt -F /root/.muttrc -s "[INTRUDER ALERT] Intruder Detected" -a $IMAGEFILE -- you@gmail.com
```

发送的邮件会有类似下面的内容。

```
Warning,
An intruder was detected at Sat Sep 15 22:50:32 EST 2012
Please see the image attached for details
```

可以看到我们通过 `date` 命令实现了系统时间输出，并通过添加“\n”配合 `echo` 命令的参数“-e”实现换行。至此我们已经积累了不少代码片段，下面我们将把它们都串起来。

脚本 1

首先，我们需要检查文件夹中文件数是否为 0。

```
if [ $NUMFILES -gt 0 ]; then
```

下面我们需要做的就是将文件夹中的图像文件逐一以附件形式发送，然后删除它们。显然我们需要使用一个循环来实现，这里使用 for 循环。

```
for IMAGEFILE in $LISTFILES
```

 **注意** 别忘了用 `chmod +x` 为脚本添加执行权限。

完整的脚本 1 如下：

```
#!/bin/bash
MOTIONCONF=/etc/motion/motion.conf
MOTIONDIR=' sudo grep "^target_dir" $MOTIONCONF | awk '{ print $2}''
if [ ! -d $MOTIONDIR ]; then
    mkdir $MOTIONDIR
    chown motion $MOTIONDIR
fi
LISTFILES=('find $MOTIONDIR -maxdepth 1 -type f | grep jpg$')
NUMFILES=' find $MOTIONDIR -maxdepth 1 -type f | grep jpg$ | wc -l'
if [ $NUMFILES -gt 0 ]; then
    for IMAGEFILE in $LISTFILES
    do
        echo -e "Warning,\nAn intruder was detected at 'date'\nPlease see the
image attached for details" \
        | mutt -s "[INTRUDER ALERT] Intruder Detected" \
        -F /root/.muttrc \
        -a $IMAGEFILE -- <your email>
        rm $IMAGEFILE
    done
fi
```

脚本 2

脚本 2 的实现比脚本 1 稍微复杂些。首先我们需要统计已经处理过的图像文件数量（变量 LASTCOUNT），然后通过一个死循环来保证脚本可以一直运行（见第 7 章）。通过变量 LASTCOUNT 和变量 NUMFILES 的比较我们就可以判断是否需要发送新的邮件。

```
while [ $LASTCOUNT -lt $NUMFILES ]
```

接下来我们需要获得图像文件的文件名，好在 motion 创建的图像文件是连续命名的。我们可以通过如下方式来获取文件名。

```
IMAGEFILE=${LISTFILES[$ITERATOR]}
```

别忘了要将累加变量加 1。

```
LASTCOUNT='expr $LASTCOUNT + 1' and ITERATOR='expr $ITERATOR + 1'
```

完整的脚本 2 如下：

```
#!/bin/bash
MOTIONCONF=/etc/motion/motion.conf
MOTIONDIR=' sudo grep "^target_dir" $MOTIONCONF | awk '{ print $2}''
if [ ! -d $MOTIONDIR ]; then
    mkdir $MOTIONDIR
    chown motion $MOTIONDIR
fi
LISTFILES='find $MOTIONDIR -maxdepth 1 -type f | grep jpg$'
NUMFILES='find $MOTIONDIR -maxdepth 1 -type f | grep jpg$ | wc -l'
LASTCOUNT=0
while true
do
    LISTFILES=('find $MOTIONDIR -maxdepth 1 -type f | grep jpg$')
    NUMFILES='find $MOTIONDIR -maxdepth 1 -type f | grep jpg$ | wc -l'
    ITERATOR=0
    while [ $LASTCOUNT -lt $NUMFILES ];
    do
        IMAGEFILE=${LISTFILES[$ITERATOR]}
        echo -e "Warning,\nAn intruder was detected at 'date'\nPlease see the image
attached for details" \
            | mutt -F /root/.muttrc \
            -s "[INTRUDER ALERT] Intruder Detected" \
            -a $IMAGEFILE -- <your email> LASTCOUNT='expr $LASTCOUNT + 1' ITERATOR='expr
$ITERATOR + 1' done sleep 1
    done
done
```

测试脚本

在运行我们的脚本前别忘了先确认 motion 是否可以正常工作。

 **注意** 可以通过 **Ctrl** + **C** 来结束脚本运行。

我们的脚本不会在屏幕上打印任何输出，去检查下你的邮箱吧（见图 10-4）。

可以看到，脚本运行正常。那么我们如何实现脚本的开机启动功能呢？我们不妨通过 cron 启动脚本 1。对于脚本 2，我们可以将它添加到 motion 的启动脚本中。

使用 root 用户执行 “crontab -e” 命令添加如下 cron 任务。

```
* * * * * /root/script1.sh
```

这样 cron 就会每分钟运行一次脚本 1，来实现文件的检查、发送和删除。

下面我们修改 motion 的启动脚本（/etc/init.d/motion）来实现脚本 2 的启动。使用 root 用户打开/etc/init.d/motion，找到 start 相应的语句。我们需要在 motion 启动后执行脚本 2，所以我们

要将“/root/script2.sh &”添加到 start-stop-daemon 后面。末尾的“&”表示在后台执行脚本。

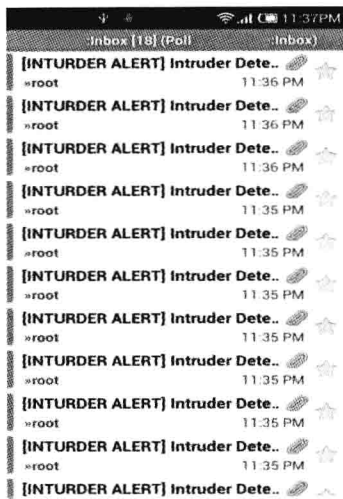


图 10-4 脚本发送的邮件

```
start)
    if check_daemon_enabled ; then
        if ! [ -d /var/run/motion ]; then
            mkdir /var/run/motion
        fi
        chown motion:motion /var/run/motion

        log_daemon_msg "Starting $DESC" "$NAME"
    if start-stop-daemon --start --oknodo --exec $DAEMON -b --chuid motion ; then
        /root/script2.sh &
        log_end_msg 0
    else
        log_end_msg 1
        RET=1
    fi
fi
;;
```

完成编辑并保存文件后重启树莓派，检查开机启动功能是否正常。如果看到类似下面的信息，说明你的程序成功启动了。

```
$ ps -ef | grep script
root      2058      1  0 11:23 pts/0    00:00:00 /bin/bash /home/pi/script2.sh
```

排错

现在你肯定深刻体会到这样一个事实，即出错在软件开发中是难以避免的。所以排错就成了

一项必不可少的技能，下面就排错给出一些小建议。

隔离调试，即独立运行程序中的特定代码段，确定其工作正常。事实上，前文中我们之所以先将代码分成一段一段就是为了方便进行排错。

使用 `echo` 命令将中间变量打印到屏幕上，确定其工作正常。

将“`#!/bin/bash`”改为“`#!/bin/bash -x`”，这样 `bash` 就会进入 `debug` 模式，并所有变量变化输出到屏幕上。

10.7 开发路线图

先来回顾下我们已经实现的功能。

- 为树莓派连接摄像头。
- 安装配置监控软件 `motion`。
- 实现运动图像捕捉。
- 安装配置 `ssmtp`。
- 安装配置 `mutt`。
- 实现通过命令行发送邮件。
- 通过两个脚本串联以上所有功能。

那么我们还可以做些什么呢？

- 添加无线网卡，这样你就可以将树莓监控派藏到角落里隐藏起来。
- 用电池替换有线电源，增加移动性。
- 保留所有图像文件，以便以后检查。
- 实现视频记录。

图 10-5 所示的就是进一步改造后的树莓监控派。

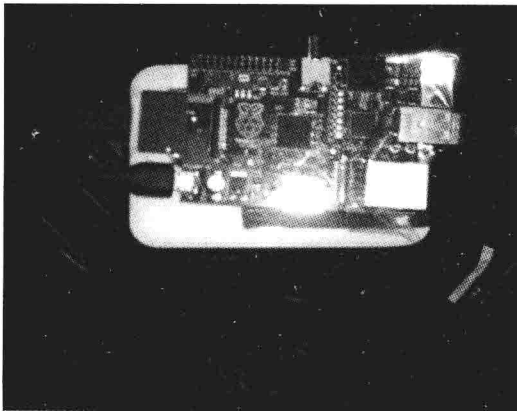


图 10-5 进一步改造后的树莓监控派

本章小结

在本章中我们学习了如何阅读 `dmesg` 和 `udev` 命令的输出结果，安装配置 `motion`、`ssmtp` 以及 `mutt`，并实现了完整的树莓监控派应用。

树莓派媒体中心

树莓派的最美妙之处在于它蕴含了无限可能，可以根据使用者的需求实现各种不同的应用。通过各种扩展接口，树莓派可以和不同的设备协同工作。

我们已经知道树莓派支持以 1920 像素 x 1080 像素的高分辨率（即 1080P）实现 HDMI 输出，而且也支持通过 HDMI 实现音频回放。在本章中我们将借助树莓派强大的媒体回放能力，搭建两种不同的树莓派媒体中心。一种用来实现网络视频播放功能，另一种实现音频流的网络分享功能。

11.1 使用 XBMC 播放视频

XBMC (XBox Media Center) 可以最大限度地解放树莓派的视频播放能力。顾名思义，XBMC 最初是为 Xbox 设计的。由于设计良好，现在 XBMC 已经被移植到各种操作系统上，诸如 Windows、Android、Mac OS X 以及 iOS。甚至有人还实现了一个专用的 Linux 发行版称为 XBMCbuntu。其开源、易用的特性也催生了大量的衍生项目，诸如 MediaPortal、Plex、Voddler 以及 Boxee 等。Boxee 甚至有称为 Boxee Box 的配套设备，售价超过 200 美元。而用树莓派我们可以用低得多的成本实现相同的功能。

在前面的章节中我们已经熟悉了如何使用 apt-get 来安装程序，通过这种方式确实可以安装大多数软件。但有部分程序我们只能通过编译源码安装，下面我们就来看看该怎么做。编译 XBMC 是十分耗时的。我坚持这样做是因为在学习编译安装的过程中可以更深入地了解 Linux 程序的构成，此外在本机编译的程序执行效率也更高。通过 apt-get 这样的包管理系统安装程序自然会比较方便，但只有学会了编译安装这样看起来较繁琐的方法才能更深入地理解系统的运行机制，这也是本书中我们一直倡导的。

首先，我们需要下载一个称为 Pisces 的 Raspbian 裁剪版 <http://www.raspbian.org/PiscesImages>。使用它是为了剪裁我们不需要的部分，最大程度地解放树莓派的视频播放能力¹。借助前面章节学习的知识，我们可以轻松地完成下载并安装镜像（需要一张至少 4GB 的 SD 卡）以及设置网络连接等操作。系统的默认用户是 raspbian，它与 root 用户的密码都是 raspbian。

¹ 译者注：简便起见，读者也可以在之前已经安装好的系统中进行下面的操作。

RASPBMC

我们也可以以一种更简单的方式在树莓派上使用 XBMC，即使用一个称为 RaspBMC 的定制版 Raspbian。读者可以在 <http://www.raspbmc.com/download/> 找到相应的镜像文件并安装。安装过程十分简单，将树莓派连接到网络后它还可以自行进行相应的下载和配置。如果你决定使用 RaspBMC，可以略过“开始使用 XBMC”这一节的内容。

注意 如果安装系统后发现找不到 `ifconfig`，可以使用“`sudo apt-get install net-tools`”进行安装（`apt-get` 的详细使用方法见第 3 章）。

在完成了操作系统的安装后我们再来一步一步地完成 XBMC 的安装。通过 `apt-get` 安装程序一般都不会太耗时，这是因为程序已经被提前编译好了，而编译安装相比之下就要花费更多的时间。另一方面由于树莓派使用了低功耗的 ARM 处理器也导致了编译时间的延长。以 XBMC 为例，整个编译过程甚至要花费 12 个小时左右也就是半天的时间。晚上睡前开始编译是个不错的主意，这样我们就可以不用盯着屏幕发呆了。

搭建编译环境

首先，我们需要确保尽可能多的内存可以用于编译操作。在 Raspbian 我们可以通过 `rapsi-config` 来实现内存的分配，由于 Pisces 下没有这一工具，我们需要手动替换 `start.elf` 文件。

```
$ sudo cp /boot/arm224_start.elf /boot/start.elf
```

这样我们就有了 224MB 的系统内存，余下的 32MB 将作为共享显存。接下来我们需要告知系统配置已修改并重启系统。

```
$ sudo rpi-update
$ sudo reboot
```

然后我们需要如下两条命令来确保系统上的程序都已经更新到最新版。执行第一行命令后，`apt-get` 工具会从软件仓库获取各种最新版的软件信息。执行第二行命令后，`apt-get` 工具会将本地已安装的软件更新到最新版。

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

下一步，我们就需要安装编译所需的工具：基础构建工具、自动配置工具、压缩 / 解压缩工具等。

```
$ sudo apt-get install build-essential autoconf ccache gawk gperf mesa-utils zip unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version.
The following packages were automatically installed and are no longer required:
```

```
libcdio-cdda0 libcdio-paranoia0 libcdio10 libcelt0-0 libdb4.8 librpmio2
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  automake autotools-dev libglew1.7 libsigsegv2
Suggested packages:
  autoconf2.13 autoconf-archive gnu-standards autoconf-doc libtool gettext distcc gawk-doc
  glew-utils
The following NEW packages will be installed:
  autoconf automake autotools-dev ccache gawk gperf libglew1.7 libsigsegv2 mesa-utils
The following packages will be upgraded:
  unzip zip
2 upgraded, 9 newly installed, 0 to remove and 187 not upgraded.
Need to get 2,647 kB/3,174 kB of archives.
After this operation, 7,143 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

完成这些基础工作后，下面要做的相对复杂些。我们需要安装一系列视频编解码库、特定的图形库以及用于远程控制 XBMC 的程序。下面这条命令非常长，我们写了一个脚本来完成相同的工作，可以在 Apress 的网站上下载到。

```
$ sudo apt-get install autotools-dev comerr-dev dpkg-dev libalsaplayer-dev \
libapt-pkg-dev libasound2-dev libass-dev libatk1.0-dev \
libavahi-client-dev libavahi-common-dev libavcodec-dev libavformat-dev \
libavutil-dev libbison-dev libbluray-dev libboost1.49-dev \
libbz2-dev libc-dev-bin libc6-dev libcaca-dev libcairo2-dev \
libcdio-dev libclalsdrv-dev libcrypto++-dev libcups2-dev libcurl3-gnutls-dev \
libdbus-1-dev libdbus-glib-1-dev libdirectfb-dev libdrm-dev libegl1-mesa-dev \
libelf-dev libenca-dev libept-dev libevent-dev libexpat1-dev libflac-dev \
libfontconfig1-dev libfreetype6-dev libfribidi-dev libgconf2-dev \
libgcrypt11-dev libgdk-pixbuf2.0-dev libgl1-mesa-dev libgles2-mesa-dev \
libglew-dev libglewmx-dev libglib2.0-dev libglu1-mesa-dev \
libgnome-keyring-dev libgnutls-dev libgpg-error-dev libgtk2.0-dev libhal-dev \
libhunspell-dev libice-dev libicu-dev libidn11-dev libiso9660-dev \
libjasper-dev libjbig-dev libjconv-dev libjpeg8-dev libkrb5-dev \
libldap2-dev libltdl-dev liblzo2-dev libmad0-dev libmicrohttpd-dev \
libmodplug-dev libmp3lame-dev libmpeg2-4-dev libmysqlclient-dev \
libncurses5-dev libnspr4-dev libnss3-dev libogg-dev libopenal-dev \
libp11-kit-dev libpam0g-dev libpango1.0-dev libpcre++-dev libpcre3-dev \
libpixmap-1-dev libpng12-dev libprotobuf-dev libpthread-stubs0-dev \
libpulse-dev librtmp-dev libsamplerate0-dev \
libSDL-image1.2-dev libSDL1.2-dev libslang2-dev \
libsm-dev libsmclient-dev libspeex-dev \
libsqlite3-dev libssh-dev libssh2-1-dev libssl-dev libstdc++6-4.6-dev \
libtagcoll2-dev libtasn1-3-dev libtiff4-dev libtinfo-dev libtinymxml-dev \
libts-dev libudev-dev libv8-dev libva-dev libvdpau-dev \
libvorbis-dev libvpx-dev libwebp-dev libwibble-dev \
libx11-dev libx11-xcb-dev libxapian-dev libxau-dev \
libxcb-glx0-dev libxcb-render0-dev libxcb-shm0-dev \
libxcb1-dev libxcomposite-dev libxcursor-dev libxdamage-dev \
```


简便起见，我们可以直接使用无参数的“cd”命令回到 home 目录。此外，如果遇到无法输出“~”符号的情况，可以通过 `dpkg-reconfigure keyboard-configuration` 来修改键位映射并重启树莓派。

通过版本控制系统 Git 所有人都可以很容易地下载最新版的源代码。

注意 最新版的源代码往往实现了各种新功能，相应的稳定性会差一些。

执行下面这条命令就可以完成 XBMC 源代码的下载。“--depth 1”表示我们只需要最新版的源代码而非之前的源代码。

```
$ git clone --depth 1 git://github.com/xbmc/xbmc-rbp.git
Cloning into 'xbmc-rbp'...
remote: Counting objects: 35172, done.
remote: Compressing objects: 100% (22895/22895), done.
remote: Total 35172 (delta 15265), reused 27885 (delta 10698)
Receiving objects: 100% (35172/35172), 158.26 MiB | 385 KiB/s, done.
Resolving deltas: 100% (15265/15265), done.
```

使用 Sed 和正则表达式修改文件

在实际编译 XBMC 时，需要按我们的需求对源代码做些小的修改。我们通过 Sed (Stream Editor) 来自动执行这些修改。Sed 可以通过正则表达式实现强大的文本查找替换操作。在第 6 章中我们简单介绍过正则表达式。很多情况下它可以快速便捷地解决复杂的问题，譬如识别一个字符串是否是电子邮件地址。我们知道邮件地址由用户名、“@”以及域名构成，比如 `username@gmail.com`。使用“`*@*`”这样简单的查询方式显然是不可行的，因为“`*`”会匹配空格以及各种符号。这类精准的匹配任务用正则表达式来做会更得心应手。正则表达式涉及的内容十分广泛，本书无法展开叙述，感兴趣的读者可以通过第 6 章中我们推荐的网站做进一步的学习。

下面两条命令实现了对 `tools/rbp/setup-sdk.sh` 的自动修改，将变量 `USE_BUILDROOT` 的值由 1 改为 0，将 `TOOLCHAIN=/usr/local/bcm-gcc` 改为 `TOOLCHAIN=/usr`。“`-i`”参数表示直接修改文件。

```
$ sed -i 's/USE_BUILDROOT=1/USE_BUILDROOT=0/' tools/rbp/setup-sdk.sh
$ sed -i 's/TOOLCHAIN=/usr/local/bcm-gcc/TOOLCHAIN=/usr/' tools/rbp/setup-sdk.sh
```

接下来，执行：

```
$ sudo sh tools/rbp/setup-sdk.sh
```

这一脚本会生成我们所需的 Makefile 文件，位于 `tools/rbp/depends/xbmc/Makefile`。我们需要使用 Sed 注释掉这一文件中所有的“`cd $(SOURCE); $(CONFIGURE)`”命令。

```
$ sed -i 's/cd $(SOURCE); $(CONFIGURE)/#cd $(SOURCE); $(CONFIGURE)/'
tools/rbp/depends/xbmc/Makefile
```

这样做是为了保证在编译过程中使用系统自带的库文件而非 XBMC 源代码中的文件。

编译源代码

多数编译操作的逻辑十分简单，即执行“./configure”根据系统情况生成用于编译的配置文件 makefile。我们这里要进行的编译更加高阶一些，需要先对配置工具做一些设置。

```
$ make -C tools/rbp/depends/xbmc/
```

在长长的输出结果后你会看到如下内容。

```
examples/Makefile.am: installing './depcomp'
Makefile.am: installing './INSTALL'
autoreconf: Leaving directory 'lib/libdvd/libdvdnav'
Please (re)run configure...
#cd ../../../../; ./configure --prefix=/opt/xbmc-bcm/xbmc-bin --build=i686-linux --host=arm-
bcm2708-linux-gnueabi --enable-gles --disable-sdl --disable-x11 --disable-xrandr --disable-openmax
--disable-optical-drive --disable-dvdcss --disable-joystick --disable-debug --disable-crystalhd
--disable-vtbdecoder --disable-vaapi --disable-udpau --disable-pulse --disable-projectm
--with-platform=raspberry-pi --disable-optimizations --enable-rpi-cec-api
#cd ../../../../; make -j 1
#cd ../../../../; make install
make: Leaving directory '/home/raspbian/xbmc-rbp/tools/rbp/depends/xbmc'
```

下面又是一条很长的配置命令，好在我们的脚本里也包含了这部分内容。它指定了安装位置（`--prefix=/usr`）、编译架构（`--build=arm-linux-gnueabi`）、库文件位置（`--localstatedir=/var/lib`）以及平台信息（`--with-platform=raspberry-pi`），并除去了一些我们不需要的功能。

```
$ ./configure --prefix=/usr --build=arm-linux-gnueabi --host=arm-linux-gnueabi \
--localstatedir=/var/lib --with-platform=raspberry-pi --disable-gl --enable-gles \
--disable-x11 --disable-sdl --enable-ccache --enable-optimizations \
--enable-external-libraries --disable-goom --disable-hal --disable-pulse \
--disable-vaapi --disable-udpau --disable-xrandr --disable-airplay \
--disable-alsa --enable-avahi --disable-libbluray --disable-dvdcss \
--disable-debug --disable-joystick --enable-mid --disable-nfs --disable-profiling \
--disable-projectm --enable-rsxs --enable-rtmp --disable-vaapi \
--disable-vddecoder --disable-external-ffmpeg --disable-optical-drive
```

当命令执行完成后，我们会看到如下结果。

```
XBMC Configuration:
Debugging:      No
Profiling:      No
Optimization:   Yes
Crosscomp.:     No
target ARCH:    arm
target CPU:     arm1176jzf-s
OpenGLES:       Yes
ALSA:           No
DBUS:           Yes
VDPAAU:         No
VA-API:         No
CrystalHD:      No
VDADecoder:     No
```

```

VTBDecoder:    No
OpenMax:       No
Joystick:      No
XRandR:        No
GOOM:          No
RSXS:          Yes
ProjectM:      No
Skin Touched:  No
X11:           No
Bluray:        No
TexturePacker: Yes
MID Support:   Yes
ccache:        Yes
ALSA Support:  No
PulseAudio:    No
HAL Support:   No
DVDCSS:        No
Avahi:         Yes
Non-free:      Yes
ASAP Codec:    No
MySQL:         Yes
Webserver:     Yes
libRTMP support:    Yes
libsmbclient support: Yes
libnfs client support: No
libafpclient support: No
AirPlay support:    No
AirTunes support:    No
Optical drive:      No
libudev support:     Yes
libusb support:      No
libcec support:      No
libmp3lame support:   Yes
libvorbisenc support: Yes
libcap support:       No
External FFmpeg:     No
prefix:               /usr

```

这样我们就完成了 XBMC 针对树莓派的优化配置。下面我们下面这条简单的命令就可以开始编译了。

```
$ make
```

看到如下的警告信息别紧张，编译过程只有遇到关键性错误才会停止。

```
/tmp/ccGvUe1g.s:507: Warning: swp{b} use is deprecated for this architecture
```

在编译结束后应该会看到如下结果。

```
XBMC built successfully
```

显然，这表明 XBMC 已经成功编译。接下来只要执行“`sudo make install`”进行安装操作就可以了。

```
$ sudo make install
Copying XBMC binary to /usr/lib/xbmc/xbmc.bin
You can run XBMC with the command 'xbmc'
Copying support and legal files...
Done!
Copying system files to /usr/share/xbmc
```

在经历了漫漫长途后我们终于完成了编译安装。虽然有些耗时，但也不算太难，不是吗？

排错

我们的多数设置是对所有树莓派都通用的，我们不妨来看看在编译过程中如何排除可能出现的错误。首先确保成功安装了相关工具，复制了指定的头文件并建立了相应的软链接。如果错误依旧存在，删除 `xbmc-rbp` 文件夹并重新开始总是能解决问题的。

在某些情况下，Git 可能会出现下载到一半卡住的情况。这时按下 `Ctrl` + `C` 停止当前操作，删除已下载的文件，并重新开始。在编译过程中移除外设减少系统负担也可以提高编译成功率。此外，确保“XBMC Configuration”的输出结果与我们给出的设置相同，可以最大程度减少编译时间。

11.2 开始使用 XBMC

我们终于完成了复杂的编译过程，这节可以准备启动 XBMC。我们首先重置之前的配置，XBMC 将会执行复杂的图形运算，所以需要多分配一些内存到图形处理器中。我们通过执行以下的命令在树莓派启动时修改内存分配。

```
$ sudo cp /boot/arm128_start.elf /boot/start.elf
```

之后检查一下之前的链接文件是否正确。我们还可以修改 `/boot/config.txt` 文件，该文件可以调整树莓派底层硬件的行为，最简单的修改是添加 `disable_overscan=1` 这一行配置，这样系统就不会进行过扫描处理，由于过扫描（overscan）会导致视频回放消耗更多的内存，这样设置后就可以节约一些系统资源。完成修改后，我们需要使用以下命令让树莓派重新读取配置并重启。

```
$ sudo rpi-update
$ sudo reboot
```

这样我们就完成了 128/128 的内存分配比例调整，启动树莓派，以 `root` 身份登录并执行以下命令。

```
$ /usr/lib/xbmc/xbmc.bin
```

树莓派的屏幕将会静止一会儿，之后会出现如图 11-1 所示的 XBMC 启动画面。



图 11-1 XBMC 启动画面

等待一会儿，就会出现 XBMC 的主菜单，如图 11-2 所示。

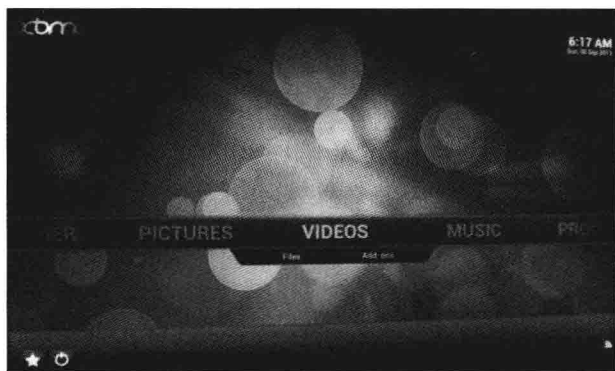


图 11-2 XBMC 主菜单

XBMC 是完全的图形化系统，在主菜单中，可以选择回放多种媒体文件。XBMC 还可以通过网络回放其他计算机共享的多媒体文件。现在我们是通过键盘对 XBMC 进行操作，但是作为媒体中心，键盘绝对不是最佳的选择，所以我们接下来为树莓派安装遥控器。有两种解决方案。

- 简单一些的方案是购买一个 USB 遥控器并安装到树莓派。
- 更有趣的方案是使用智能手机作为遥控器。

大多数的智能手机都可以安装一款 XBMC 的遥控器应用，在 XBMC 上做一些简单的设置之后，就可以通过你的智能手机进行遥控了。我们在 XBMC 的设置菜单选择 Services（服务），调整 remote control settings（遥控器设置）选项，选择“Allow programs on other systems to control XBMC”（允许其他计算机上的程序控制 XBMC）。之后在 Webserver（网络服务）中选择“Allow control of XBMC via HTTP”（允许通过 HTTP 控制 XBMC），打开该选项后，XBMC 会要求设置用户名和密码，以提高安全性。

完成了 XBMC 上的设置后，我们需要做的就是下载并安装遥控应用。启动遥控应用后，会看到如图 11-3 所示的界面，填写你的树莓派 IP 地址（在 XBMC 的 System>System info 菜单中查看），将端口设置为 8080，填写之前设置的用户名和密码。

完成了应用的设置后，你的智能手机就变成了 XBMC 的遥控器，界面上会出现各种操控按钮，如图 11-4 所示。

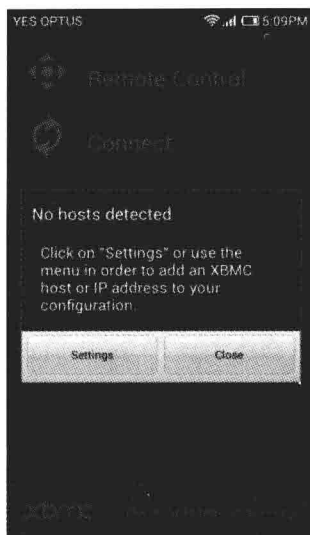


图 11-3 XBMC 遥控器应用



图 11-4 XBMC 遥控器开始工作

自动启动 XBMC

我们已经能够在树莓派上手动启动 XBMC 了，不过作为媒体中心，XBMC 能够自动启动会比较方便，否则每次都需要连接键盘输入一些命令才能使用。

本节我们编写一段 start 脚本，这样系统启动时将通过 start 自动启动 XBMC。我们创建 /etc/init.d/xbmc 文件，并输入以下内容。

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          xbmc
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start XBMC
# Description:       Start XBMC
### END INIT INFO
DAEMON=/usr/bin/xinit
DAEMON_OPTS="/usr/lib/xbmc/xbmc.bin"
NAME=xbmc
DESC=XBMC
RUN_AS=root
PID_FILE=/var/run/xbmc.pid
test -x $DAEMON || exit 0
```

```

set -e
case "$1" in
  start)
    echo "Starting $DESC"
    start-stop-daemon --start -c $RUN_AS --background --pidfile $PID_FILE --make-pidfile
--exec $DAEMON -- $DAEMON_OPTS
    ;;
  stop)
    echo "Stopping $DESC"
    start-stop-daemon --stop --pidfile $PID_FILE
    ;;
  restart|force-reload)
    echo "Restarting $DESC"
    start-stop-daemon --stop --pidfile $PID_FILE
    sleep 5
    start-stop-daemon --start -c $RUN_AS --background --pidfile $PID_FILE --make-pidfile
--exec $DAEMON -- $DAEMON_OPTS
    ;;
  *)
    echo "Usage: /etc/init.d/$NAME{start|stop|restart|force-reload}" >&2
    exit 1
    ;;
esac
exit 0

```

完成后，我们保存启动脚本，执行以下命令将 XBMC 加入树莓派启动过程。

```

$ sudo chmod +x /etc/init.d/xbmc
$ sudo update-rc.d xbmc defaults

```

重启树莓派之后，XBMC 会自动启动，这样我们就完成了 XBMC 的基本设置。

排错

本章，我们从源代码开始，手工编译了 XBMC，所以如果 XBMC 的编译和安装过程都全部成功，那么应该不会出现运行问题。不过，XBMC 在启动时有可能出现以下警告。

```
libEGL warning: DRI2: xcb_connect failed
```

该警告是由于库文件错误造成的，如果出现了该警告，可以通过重新执行之前的拷贝和链接命令解决。

11.3 使用 AirPlay 播放音乐

这节我们将让树莓派变成一个网络流媒体播放器，完成后你就可以用音响系统连接树莓派，使用遥控器播放音乐。目前有很多网络回放系统，本节我们将使用 Apple 的 AirPlay 协议。需要说明的是，本节后面的内容同样适用于原版的 Raspbian 系统和 Pisces 系统。

为树莓派实现 AirPlay 播放能力比安装 XBMC 要简单得多,虽然还是需要做一些源代码编译,不过我们基本不需要使用 `./configure` 进行配置。为了测试流媒体播放能力,请先连接好音箱或音响系统。

我们先安装一些软件包。

```
$ sudo apt-get install build-essential libssl-dev libcrypto-openssl-rsa-perl libao-dev
libio-socket-inet6-perl libwww-perl avahi-utils pkg-config alsa-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version.
libio-socket-inet6-perl is already the newest version.
libio-socket-inet6-perl set to manually installed.
libssl-dev is already the newest version.
libwww-perl is already the newest version.
libwww-perl set to manually installed.
pkg-config is already the newest version.
pkg-config set to manually installed.
The following packages were automatically installed and are no longer required:
  libcdio-cdda0 libcdio-paranoia0 libcdio10 libcelt0-0 libdb4.8 librpmio2
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  alsa-base avahi-daemon libao-common libao4 libavahi-core7 libcrypto-openssl-bignum-perl
libdaemon0 libnss-mdns
Suggested packages:
  alsa-oss oss-compat avahi-autoipd libaudio2 libesd0 libesd-alsa0
The following NEW packages will be installed:
  alsa-base alsa-utils avahi-daemon avahi-utils libao-common libao-dev libao4 libavahi-core7
libcrypto-openssl-bignum-perl libcrypto-openssl-rsa-perl libdaemon0
libnss-mdns
0 upgraded, 12 newly installed, 0 to remove and 139 not upgraded.
Need to get 1,699 kB of archives.
After this operation, 3,702 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

安装完成后,需要载入音频驱动模块。我们可以使用 `modprobe` 命令载入驱动模块,选择载入 `snd_bcm2835`,对应于树莓派搭载的 Broadcom 音频芯片。

```
$ sudo modprobe snd_bcm2835
```

该命令执行后不会有任何输出,所以我们需要通过另一个命令确定模块是否载入了系统。我们介绍过 `ls` 命令用于列出文件和目录, `lsusb` 命令用于列出 USB 设备,类似的, `lsmod` 命令可以列出系统中正在使用的模块。我们执行 `lsmod`,检查 `snd_bcm2835` 模块是否已经载入系统。

```
$ lsmod
Module                               Size  Used by
snd_bcm2835                          21485  0
```

```

snd_pcm                82208  1 snd_bcm2835
snd_page_alloc         5383   1 snd_pcm
snd_seq                59808   0
snd_seq_device         6920   1 snd_seq
snd_timer              21905   2 snd_seq,snd_pcm
snd                    57668   5 snd_timer,snd_seq_device,snd_seq,snd_pcm,snd_bcm2835
iprv6                  290227 34
r8712u                 182646  0
spi_bcm2708            4815   0
i2c_bcm2708            3818   0

```

由于是首次载入音频模块，我们还需要设置音频设备，执行 `amixer` 命令。

```

$ sudo amixer cset numid=3 1
numid=3,iface=MIXER,name='PCM Playback Route'
; type=INTEGER,access=rw-----,values=1,min=0,max=3,step=0
: values=1

```

这里我们将 `numid=3`（PCM 回放）输出至接口 1，也就是 3.5mm 音频接口；如果你选择接口 2，则可以将 PCM 回放输出至树莓派搭载的 HDMI 音频接口。

完成硬件和驱动的设置，下一步我们从网络下载并安装 `shairport` 软件。我们使用 `wget` 命令（Web GET 的缩写）下载完整的 `shairport` 源代码。

```

$ wget https://github.com/albertz/shairport/zipball/master
--2012-09-30 18:13:36-- https://github.com/albertz/shairport/zipball/master
Resolving github.com (github.com)... 207.97.227.239
Connecting to github.com (github.com)|207.97.227.239|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nodeload.github.com/albertz/shairport/zipball/master [following]
--2012-09-30 18:13:42-- https://nodeload.github.com/albertz/shairport/zipball/master
Resolving nodeload.github.com (nodeload.github.com)... 207.97.227.252
Connecting to nodeload.github.com (nodeload.github.com)|207.97.227.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46413 (45K) [application/zip]
Saving to: 'master'
100%[=====>] 46,413      60.1K/s   in 0.8s
2012-09-30 18:13:50 (60.1 KB/s) - 'master' saved [46413/46413]

```

下载的 `shairport` 源代码包名为 `master`，我们使用 `mv` 命令（move 的缩写）将其重命名，并使用 `unzip` 命令解压缩该源代码包，最后使用 `cd` 进入目录，准备编译。

```

$ mv master albertz-shairport-b58f156.zip
$ unzip albertz-shairport-b58f156.zip
$ cd albertz-shairport-b58f156

```

接下来就是编译 `shairport`，该过程远比编译 XBMC 简单。它无需执行编译配置，直接执行 `make` 即可。

```

$ make
cc -O2 -Wall -c alac.c -o alac.o
cc -O2 -Wall -DHAIRTUNES_STANDALONE hairtunes.c alac.o -o hairtunes -lm -lpthread -lssl

```

```

-lcrypto -lao
cc -O2 -Wall -c socketlib.c -o socketlib.o
cc -O2 -Wall -c shairport.c -o shairport.o
cc -O2 -Wall -c hairtunes.c -o hairtunes.o
cc -O2 -Wall socketlib.o shairport.o alac.o hairtunes.o -o shairport -lm -lpthread -lssl
-lcrypto -lao

```

完成编译后，我们运行 shairport.pl 测试一下 shairport。

```

$ ./shairport.pl
Can't locate Net/SDP.pm in @INC (@INC contains: /usr/lib/perl5/site_perl /usr/share/perl5/site_perl
/usr/lib/perl5/vendor_perl /usr/share/perl5/vendor_perl /usr/lib/perl5/core_perl /usr/share/perl5/
core_perl .) at ./shairport.pl line 45.

```

这里出现了一个错误，运行该 Perl 脚本需要 Net/SDP 模块，而树莓派并没有自带该模块，所以需要安装它。

由于无法使用 apt-get 命令自动安装 Perl 模块，我们需要从 Perl CPAN 库（CPAN 库类似于 apt-get 的软件源，其中包含大量 Perl 模块）下载该模块，并手动安装它。再次执行 wget 命令下载文件。

```

$ wget http://search.cpan.org/CPAN/authors/id/N/NJ/NJH/Net-SDP-0.07.tar.gz
--2012-09-30 19:01:11-- http://search.cpan.org/CPAN/authors/id/N/NJ/NJH/Net-SDP-0.07.tar.gz
Resolving search.cpan.org (search.cpan.org)... 199.15.176.161
Connecting to search.cpan.org (search.cpan.org)|199.15.176.161|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://mirror.westfield.com.au/cpan/authors/id/N/NJ/NJH/Net-SDP-0.07.tar.gz [following]
--2012-09-30 19:01:11-- http://mirror.westfield.com.au/cpan/authors/id/N/NJ/NJH/Net-SDP-0.07.tar.gz
Resolving mirror.westfield.com.au (mirror.westfield.com.au)... 203.42.62.21
Connecting to mirror.westfield.com.au (mirror.westfield.com.au)|203.42.62.21|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20679 (20K) [application/x-gzip]
Saving to: 'Net-SDP-0.07.tar.gz'
100%[=====>] 20,679
--.-K/s in 0.05s
2012-09-30 19:01:12 (382 KB/s) - 'Net-SDP-0.07.tar.gz' saved [20679/20679]

```

下载完成后，我们使用 tar 命令和-zxvf 参数解压该文件，之后进入 Net-SDP-0.07 目录。

```

$ tar -zxvf Net-SDP-0.07.tar.gz
$ cd Net-SDP-0.07

```

解压完成后，调用 Perl 编译系统，编译安装模块。首先运行 Build.PL 文件，生成编译文件。

```

$ perl Build.PL
Created MYMETA.yml and MYMETA.json
Creating new 'Build' script for 'Net-SDP' version '0.07'

```

之后执行编译。

```

$ ./Build
Building Net-SDP

```

编译完成后，使用 test 参数执行模块内置的测试用例，确保编译的模块没有错误。

```

$ ./Build test
t/00use.t ..... ok
t/10generate.t .. ok
t/10parse.t ..... ok
t/20repeat.t .... ok
t/30asstring.t .. ok
All tests successful.
Files=5, Tests=69, 5 wallclock secs ( 0.97 usr 0.05 sys + 3.40 cusr 0.29 csys = 4.71 CPU)
Result: PASS

```

最后，我们使用 build 脚本的 install 命令安装该模块。

```

$ sudo ./Build install
[sudo] password for raspbian:
Building Net-SDP
Installing /usr/local/man/man1/sdp2rat.1p
Installing /usr/local/share/perl/5.14.2/Net/SDP.pm
Installing /usr/local/share/perl/5.14.2/Net/SDP/Time.pm
Installing /usr/local/share/perl/5.14.2/Net/SDP/Media.pm
Installing /usr/local/man/man3/Net::SDP::Media.3pm
Installing /usr/local/man/man3/Net::SDP::Time.3pm
Installing /usr/local/man/man3/Net::SDP.3pm
Installing /usr/local/bin/sdp2rat

```

模块安装完成，我们再次运行 shairport.pl 脚本，看看其是否可以正常工作。

```

$ cd..
$ ./shairport.pl
Established under name '891BEA3BF8A1@ShairPort 2113 on pisces'

```

运行 shairport 后，我们尝试在 iTunes 中连接树莓派（见图 11-5）。连接树莓派上的 ShairPort，选择一首你喜欢的歌曲并播放。如果音箱播放出了音乐，那我们就成功了。

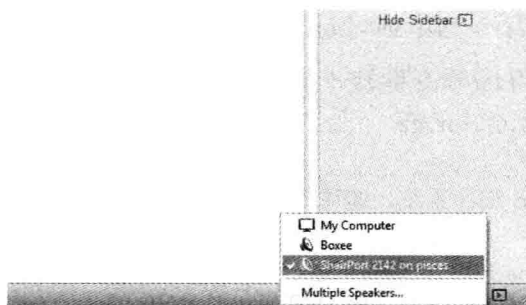


图 11-5 连接 ShairPort

排错

由于树莓派的硬件都是一样的，所以本章的所有命令都应该能够直接运行。如果你的音箱没有放出音乐，请检查一下 modprobe 命令和 amixer 命令是否正确执行。如果仍然不能解决，请重

新执行所有的配置。

自动启动 Airport

我们之前已经设置了自动启动 XBMC，同样也可以设置自动启动 shairport，这样每次启动树莓派后，就可以直接播放流媒体内容了。要做到自动启动 shairport 很容易，因为它自带了所需的文件，我们只需要执行 `sudo make install` 即可。

```
$ sudo make install
install -D -m 0755 hairtunes /usr/local/bin/hairtunes
install -D -m 0755 shairport.pl /usr/local/bin/shairport.pl
install -D -m 0755 shairport /usr/local/bin/shairport
```

完成安装后，我们将 `init` 脚本拷贝到 `init.d` 目录中。

```
$ sudo cp shairport.init.sample /etc/init.d/shairport
```

完成拷贝后，我们还需要对脚本做一些小小的修改，将 `modprobe` 命令加入其中，这样音频模块就会在每次启动时载入系统。我们打开之前拷贝的 `/etc/init.d/shairport` 文件，在 `start` 函数中增加一行 `modprobe snd_bcm2835`。

```
start() {
    echo -n "Starting shairport: "
    modprobe snd_bcm2835
    start-stop-daemon --start --quiet --pidfile "$PIDFILE" \
        --exec "$DAEMON" -b --oknodo -- $DAEMON_ARGS
    log_end_msg $?
}
```

完成后我们执行下面的命令启用 shairport 自动启动脚本。

```
$ sudo chmod +x /etc/init.d/shairport
$ sudo update-rc.d shairport defaults
```

这两条命令为 shairport 的启动脚本增加可执行权限¹，之后将其加入启动过程。我们重启树莓派后，shairport 将会自动启动。

本章小结

本章包含了很多内容，我们从头开始安装 XBMC，下载源代码并配置编译，最后花了近 12 个小时编译出了 XBMC。

我们还介绍了如何编写 `start` 脚本，并且配置了自动启动 XBMC 和流媒体服务。

最后，我们终于可以躺在沙发上，用智能手机作为遥控器控制树莓派播放音乐了！本书也到此结束。

¹ 译者注：原文内容是“增加可写权限”，但命令实际是增加可执行权限。