# 第3章　网络硬件的配置

迄今为止，关于网络接口和常见的 TCP/IP问题，我们已谈了不少，但尚未真正接触到内核中的“ 联网程序 ”访问硬件时所发生的事。鉴于此，还必须为大家讲讲接口和驱动程序这两个概念。

首当其冲的是硬件本身。比如以太网卡：它是一片环氧基树脂卡，上面布满了微晶片，这些微晶片上还有些编号，这块卡插在计算机内的一个插槽内。我们通常称之为设备（device）。

如果希望能够使用以太网卡，你必须在自己的内核中准备一些特殊的功能，使之能识别这种设备特有的访问方式。这就是所谓的设备驱动程序。例如， Linux中就有几个以太网卡驱动程序，这几个程序的功能都差不多。其中，最有名的是 Becker串行驱动程序（得名于其作者Donald Becker）。另一个是D-Link驱动程序，该程序对附着在一个并行端口上的 D-Link封装适配器进行控制。

在提到驱动程序“ 控制 ”设备时，其含义究竟是什么？首先回头看看上面提到的以太网卡。驱动程序必须能够与卡上的外设进行通信：它必须向卡发送命令和数据，而卡也应该将驱动程序发来的所有数据统统投递出去。

PC中，这种通信常常发生在一个 I/O内存区内，该内存区对应板载寄存器。内核发送给卡的所有命令和数据都必须通过这些寄存器。 I/O内存区一般被描述为起点或基础地址。以太网卡的典型基础地址是0x300或0x360。

通常情况下，不要去在意基础地址之类的硬件问题，因为内核会在启动时，对设备位置进行侦测。这就是所谓的“ autoprobing ”（自动侦测），意思是如果已安装特定的以太网卡，内核就会对若干个内存位置进行读取，并把它所读取的数据和它看到的数据进行比较。但是，也有内核不能自动侦测的以太网卡；比如，一些便宜仿造标准网卡的以太网卡。另外，内核在启动时，只能试着侦测一个以太网设备的位置。如果你使用的以太网卡不止一个，就必须清楚地将这些网卡的情况告诉内核。

另一个必须告诉内核的参数是中断请求通道( interrupt request channel)。有的硬件组件在特别需要重视时，通常可能中断内核。比如，数据抵达或出现特殊的情况。在 PC中，15个中断通道（编号0、1、3一直到15）中，其中之一可能会发生中断。分配给硬件组件的中断编号叫作“ 中断请求编号 ”或 IRQ（IRQ2和9是一样的，因为PC有两个层叠式中断处理器，每个处理器都有8个IRQ；辅助处理器连接的是主处理器的 IRQ 2）。

正如我们在第1章中所讲的那样，内核通过一个所谓的接口访问设备。接口提供了适用于所有硬件的一个抽象功能集，比如收发数据报。

接口的识别是通过接口名进行的。接口名是在内核内部定义的，而不是 /dev目录下的设备文件。常见的接口名用于以太网接口的 eth0、eth1等等。为设备分配接口常常和设备的配置顺序有关；比如，第一块以太网卡是eth0，下一个将是eth1，以此类推。唯一例外的是SLIP接口，它是动态分配的；也就是说，只要一建立 SLIP链接，就会为串行端口分配一个接口。内核将

在启动时，显示它所侦测的设备和它所安装的接口。

## 3.1 内核配置

在运行一个系统时，应该对内核的构建非常熟练。这方面的基础知识可参见马特·维尔希所著的《安装和入门指南》（这本指南也包括在 Coriolis Group的《Linux系统编程白皮书》内）。本小节，我们只为大家讨论一些连网所涉及的配置选项。

在运行make config时，首先会要求你回答几个常见的配置问题，比如，是否希望内核数学模拟等等。其中之一是问你是否需要 TCP/IP支持。必须回答“ Y ”（是），才能获得内核连网能力。

### 3.1.1 内核选项1.0及以上版本

**注意** 本小节无示例。要查找更新内容，请参考在线版。

结束常见配置询问之后，配置会继续问一些不同特性方面的问题，比如 SCSI驱动程序等。接下来的问题仍然和连网支持有关。由于 Linux的开放性，要想完整地罗列出所有的配置选项几乎是不可能的。不过， 1.0到1.1之间的大多数内核选项版本都提供了一份常见选项清单（加引号部分是批注）：

如果你想使用“ 任何 ”类型的连网设备（不管它是以太网，是 SLIP还是PPP）时，不管扩弧内显示什么样的宏名，都必须回答“ Y ”（是）。如果回答“ Y ”（是），就可以自动启用对以太网设备的支持。对其他类型网络驱动程序的支持则必须单独启用。

这些问题和Linux支持的不同链路层协议有关。 SLIP允许你通过串行线路传输 IP数据报。压缩报头（compressed header）选项提供了对CSLIP的支持，这种压缩技术将TCP/IP报头压缩为三个字节。注意，这个内核选项没有自动打开 CSLIP；它只是为CSLIP提供了必要的内核功能。

PPP是通过串行线路发送网络通信的另一种协议。它比 SLIP更为灵活，对IP也没有什么限制，同时还支持IPX。PLIP为通过并行连接发送IP数据报提供了一种解决办法。主要用于与运行DOS的计算机进行通信。

接下来的问题则和不同厂家的以太网卡有关。由于新的驱动程序层出不穷，这方面的问题肯定也是有增无减。如果想建立一个适用于不同类型机器的内核，可以采用多个驱动程序。

最后，是文件系统，配置脚本将问你是否想用 NFS（连网文件系统）。NFS会令你将文件系统导向若干台主机，使其类似于附在主机上的普通硬盘文件。

### 3.1.2 内核选项1.1.14及以上版本

**注意** 本小节无示例。

从1.1.14开始，由于增加了对IPX的Alpha支持（处于测试阶段），配置过程有了少许变化。常见选项这部分将问你是否需要常规连网支持。随后，是涉及到各种连网选项的两个问题。

要想采用TCP/IP连网，必须回答“ Y ”（是）。但回答“ N ”（否）的话，也能编译具有IPX支持的内核。

如果你的系统是两个以太网或一个以太网和一个 SLIP之间的网关，就必须启用这个选项。

尽管通过默认设置启用它没什么坏处，但你肯定想取消这一选项，把一台主机配置为一个所谓的防火墙。防火墙即是连接两个或两个以上网络，但不会在这些网络间路由通信的主机。它们常用于对来自公司网络的用户提供因特网访问，保护公司内部网络不受来自因特网的攻击和破坏。用户将得到许可登录到防火墙，使用因特网服务，但公司的机器不会因此而受到外界的攻击和破坏。因为任何接入的连接是不能通过防火墙的。

这个选项和PC/TCP的某些版本不兼容，后者是针对基于 DOS的计算机的一种商业 TCP/IP 实施方案。如果启用了这个选项，虽然仍然可以和普通计算机进行通信，但性能肯定会大受影响。

这个选项的作用是启用了 RARP（逆向地址解析）。无盘客户机和 X终端在启动时，利用 RARP来查询自己的IP地址。只有计划充当这类客户机时，才有启用 RARP的必要。网络公用程序的最新封装（net-0.32d）中，包含了一个小型的公用程序，其名为 rarp，它允许在RARP缓存内增添系统。

在通过TCP链路发送数据时，内核必须在把数据交给 IP协议之前，将它分为若干个包。对通过本地网络（比如以太网）就能抵达的主机来说，可采用较大的数据包（相对于必须通过长距离链路才能抵达的主机而言）。这样可避免小型数据包通过链路之后产生的碎片。如果不启用SNARL，内核将事实上只有一个接口的网络假定为本地网络。看看 Groucho Marx大学的网络B，整个网络都是本地的，但多数主机只连接了一个或两个子网。如果启用 SNARL，内核就会假定“所有”的子网都是本地的，在与校园内的其他所有主机通信时，都会发送大型数据包。

如果想对发往某些特殊主机（比如这种情况：数据将通过 SLIP链路）的数据采用小型数据包的格式，利用路由的mtu选项即可。关于mtu（最大传输单元）的详情，参见上一章。

对避免发送特别小的 IP包（也称作 tinygrams）来说， Negle算法是颇有启发性的。tinygrams（微型豆）通常由一些交互性的联网工具创建，这些工具只传输一个单独的键击，比如telnet和rsh。在诸如 SLIP之类的低带宽链路上，微型豆特别浪费带宽。某些情况下，Negle算法通过简单限制TCP数据传输的方式，试着避开这些微型豆。如果你碰上包丢失这一严重问题时，取消这一算法即可。

从1.1.16版本的内核开始， Linux对另一种驱动程序类型提供了支持，它就是伪驱动程序（dummy driver）。下面的问题将在设备驱动程序部分的开始处出现。

伪驱动程序的作用不大，但对单机或SLIP主机来说，它的用处就多了。它基本上是一个经过改头换面的回送接口（loopback）。出现这类接口的原因之一是在采用 SLIP，但没有以太网的主机时，人们希望有一个接口能一直保存自己的IP地址。有关详情，我们将在第5章讨论。

## 3.2 网络设备指南

内核针对不同类型的配置提供了大量的硬件设备驱动程序。本小节将简要介绍一些常见的驱动程序，以及用于这些驱动程序的接口名。

Linux中，有许多标准接口名（见下文）。大多数驱动程序支持的接口不止一个，所以这种情况下，就应该采用接口编号，比如 eth0、eth1等等。

■ lo 本地回送接口。和两个网络应用程序一起，供测试之用。它的运行类似于一个封闭式回路，任何写入这个接口的数据报都将立即返回本地主机的网络层。内核中始终都有一个

回送接口。

■ ethn n-th以太网接口。这是大多数以太网卡采用的一般性接口名。

■ dln 此类接口对D-Link DE-600包适配器（另一种以太网设备）进行访问。在衍生于并行端口的DE-600中，它是比较特殊的一种。

■ sln n-th SLIP接口。SLIP接口按照分配顺序，依次和串行线路对应起来。比如说，为SLIP配置的第一条串行线路就是sl0，第二条就是sl1，以此类推。内核能支持的SLIP接口多达四个。

■ pppn n-th PPP接口。和SLIP接口一样，PPP接口在串行线路转换为PPP模式之后，立即就和它对应起来。其时，内核能支持的接口多达四个。

■ plipn n-th PLIP接口。PLIP在并行线路上传递IP数据报。最多能支持三个PLIP接口。这些接口是在系统启动时，由PLIP驱动程序分配的，它们和并行端口一一对应。

对于将来可能增加的其他接口，比如 ISDN和AX.25，还可能引入其他接口名。用于 IPX（Novell连网协议）和AX.25 （供火腿无线电爱好者使用）的驱动程序尚处于开发阶段，但开始进入Alpha测试阶段了。

随后的几个小节中，我们将就上面提到的驱动程序展开讨论。

## 3.3 以太网安装

目前的网络程序对不同类型的以太网卡提供了广泛的支持。大多数驱动程序都是 Donald Becker（becker@cesdis.gsfc.nasa.gov）编写的，他为基于国家半导体 8390芯片的网卡编写了一系列驱动程序；即颇有名气的 Becker驱动程序系列。另外，还有两个产品是面向 D-Link的，其中的D-Link包适配器允许通过一个并行端口，访问以太网设备。针对这一用法的驱动程序是Bjørn Ekwall（bjOrn@blox.se）编写的。 DEPCA驱动程序则是 David C.Davies˘˘（davies@wanton.lkg.dec.com）编写的。

### 3.3.1 以太网接缆

如果你是生平第一次安装以太网卡，就有必要先了解一下布线方面的知识。以太网对布线是相当挑剔的。线缆的两段必须各有一个 50欧姆的电阻器，而且不能有任何分支（比方说，三条线缆组成一个星型连接）。如果利用一条带有 T型BNC连接头的细同轴线缆，就应该把这些连接头直接拧在网卡的连接器上，而不是插入一段线缆。

如果连接的是粗缆，就必须通过一个收发器（有时也称作以太网附单元）附上你的主机。可直接将收发器插入网卡上的 AUI端口，但也可采用一条屏蔽双绞线。

### 3.3.2 已获支持的网卡

下面将为大家列举一些广为人知的、已获 Linux支持的网卡。其完整列表在 HOWTO里面，其数目大约是这里列出的三倍之多。但是，即使你在这里的列表内看到了自己使用的网卡型号，但还是建议大家查看 HOWTO；HOWTO中列举的内容更为翔实、更为重要。需要注意的是，有些基于DMA的以太网卡使用的DMA通道和Adapter 1542 SCSI控制器默认状态下使用的通道一样。除非你亲自将其中之一移入另一个 DMA通道，否则的话，以太网卡就会把包数据写入你的硬盘专区内。

■ 3Com公司的EtherLink —— 获得支持的有3c503和3c503/16。3c507和3c509也如此。虽然Linux也支持3c501，但其速率太慢，不建议购买。

■ Novell公司的Eagle —— 获得支持的有NE1000和NE2000及其大量的克隆产品。NE1500和NE2100已获得支持。

■ Western Digital/SMC —— 获得支持的有WD8003和WD8013（SMC Elite和SMC Elite Plus）。另外，Linux还新增了对SMC Elite 16 Ultra的支持。

■ Hewlett Packard —— 获得支持的有HP 27252和HPJ27247B和HPJ2405A。另外还有D-Link DE-600包适配器DE-100、DE-200和DE-220-T。除此以外，还有一个用于 DE-650-T（一种PCMCIA卡）的补丁工具。

■ DEC —— 获得支持的有DE200(32k/64k) DE2O2、DE100和DEPCA rev E。

■ Allied Teliesis —— 已获Linux支持的有AT1500和AT1700。

要在Linux下使用上面列举的网卡，必须采用以上产品的主要分销商提供的一个预编译内核。这些产品一般含有内置驱动程序。但是，从长远的角度来看，最好用你自己的内核，编译真正能满足自己需要的驱动程序。

### 3.3.3 以太网自动侦测

系统启动时，以太网程序将试着找到网卡的位置，并判断它的类型。自动侦测代码存在两大局限。其一是，不能对所有的网卡进行准备识别。这不仅表现在一些便宜的仿造品上，还表现在WD80x3网卡上。其二是内核不能同时侦测多块网卡。因为它会假定你打算控制网卡和接口的分配问题。

如果你正在使用多块网卡，或自动侦测不能侦测你的网卡时，必须显式告诉内核该网卡的基础地址和设备名。

Net-3中，可通过两个不同的方案来完成上述任务。其一是改变或增加 drivers/net/Space.c文件的信息，该文件位于包含所有驱动程序信息的内核源代码内。不过，这一方案的前提是你对连网代码相当熟悉。第二种方案好的多，即在系统启动时，为内核提供驱动程序信息。如果用lilo来启动系统，通过lilo.conf内的append（添加）选项，指定一些参数之后，便可将这些参数传给内核。要将一个以太网设备的信息通知给内核，传递下面的参数即可：

```
ether=irq.base addr.param1.param2.name
```

前四个参数是数字化的，最后一个参数则是设备名。所有数字化的值都是可选的；如果都被省略或都设为零，内核就会试着通过侦测参数值的方式，找到这个值，或使用默认值。

第一个参数设置的是分配给这个设备的 IRQ。默认情况下，内核将试着自动侦测这个设备的IRQ通道。3c503驱动程序有一个非常特殊的特点，它从列出的 5、9、3、4中选出一个IRQ，并把网卡配置成使用这一行参数。

base_addr参数给出了网卡使用的I/O基础地址；零值表示内核将对上面列出的地址进行侦测。

至于其余两个参数，不同的驱动程序采用的方式是不一样的。对共享内存的网卡来说，比如WD80X3，它们指定了共享内存的起始点和终止点。其他网卡常用 param1来设置即将显示的调试信息级别。其值如果在 1到7之间，则表示冗余级不断上升，而如果是 8，就会把所有冗余都关掉；0表示默认设置。3c503驱动程序利用param2选定内部收发器（默认设置）或外部收发器（如果该值为1的话）。前者采用网卡的BNC连接器，后者采用它自己的AUI端口。

如果有两张以太网卡，可以自动侦测一张，将第二张卡的参数随 lilo一起传递出去。但是，必须保证驱动程序不会意外地先找到第二张卡，不然另一张卡根本没有注册的机会（也就是说根本不认你有两张网卡）。这是通过传递 lilo，一个保留选项来完成的，它显式告诉内核避免侦测第二张卡占用的I/O空间。

例如，要在作为eth1的0x300安装第二张以太网卡，需要向内核传递下列参数：

```
reserve=0x300,32 ether=0,0x300,eth1
```

reserve（保留）选项可保证内核在侦测某一网卡时，驱动程序不会去访问该网卡占用的I/O空间。另外，还可以用内核参数来改写 eth0的自动侦测：

```
reserve=0x340,32 ether=0,0x340,eth0
```

要关掉自动侦测，可将base_addr参数指定为-1：

```
ether=0,-1，eth0
```

## 3.4 PLIP驱动程序

PLIP代表并行线路IP，如果只需连接两台机器的话，这倒是最经济的连网方式。只须一个并行端口和一条特殊的线缆，便可获得 10Kbps到20Kbps的传输速率。

PLIP起初是Crynwr公司开发的。其设计思路相当高明（如果你愿意，也可把它称为hackish）：长期以来，个人电脑上采用的并行端口始终是单向的打印机端口，也就是说，要把数据从个人电脑传到某个外设，只能用 8条数据线路，除此以外，没有别的办法。 PLIP有效地解决了这一困扰，它将并行端口的 5条状态线路用作输入，并限定它们只能将所有数据当作半位元组来传输。这一操作模式就是所谓的零 PLIP模式。如今，这些单向并行端口很少有人用。因此，就有了PLIP扩展的产生，其名为模式1，它使用的是完整的8位接口。

目前， Linux只提供了对模式 0的支持。和早期的 PLIP代码不一样，现在它正试着与Crynwr的PLIP实施和NCSA Telnet中的PLIP驱动程序兼容（NCSA Telnet是一个DOS版本的常见程序，它在以太网或 PLIP线路上运行TCP/IP，而且还支持 Telnet和FTP）。要想利用PLIP连接两台机器，需要一种特殊的线缆，比如 Null Printer和Turbo Laplink线缆。但是，自己动手做一条这样的线缆并不费劲。详情参见第 19章。

参与PLIP驱动程序设计的人数不胜数。目前，它的维护主要由 Niibe Yutaka负责。如果把它编入内核，它就会为每个可能的打印机端口设置一个网络接口， plip0对应并行端口 lp0，plip1对应lp1，以此类推。

如果采用别的方式配置自己的打印机端口，必须对内核源代码的 drivers/net/Space.c内的这些值进行修改，并建立一个新的内核。

但是，这并不意味着你不能像往常那样使用这些并行端口。只有为其配置相应的接口，PLIP驱动程序才能对这些端口进行访问。

## 3.5 SLIP和PPP驱动程序

SLIP和PPP广泛用于在串行链路上发送 IP包。大量的机构为接入因特网的机器提供了拨号SLIP和PPP访问。因此，也为私人之间的 IP连接提供了可能（虽然有时可能得不偿失）。

要想运行SLIP或PPP，不必更换硬件设备；任何串行端口都可以用。由于串行端口的配置不只是针对TCP/IP连网的，所以我们单独为它设立了一章（即第4章）。有关详情可参考第4章。