第7章　点到点协议

和SLIP一样，PPP也是通过串行链接收发数据报时采用的协议，但解决了前者存在的两大不足，它让通信双方自行对诸如启动时的IP地址、最大传输单元等选项进行协商，并提供客户机验证。针对每个功能，PPP都有一个单独的协议。下面，我们将简要介绍一下PPP协议的基本构成。如果想进一步了解PPP，强烈推荐大家参阅RFC-1548中的PPP规格，以及相关的参考丛书。

PPP的底层是“高级数据链路控制协议”（High-Level Data Link Control Protocol, HDLC）。HDLC定义了单帧PPP的边界，并提供了一个16位的校验和。与更为原始的SLIP封装模式相反，PPP帧能够保存自其他协议（IP协议除外）包，比如Novell的IPX协议和Apple Talk协议。PPP协议是怎样做到这一点的呢？答案是：在基本 HDLC帧内增添一个协议字段。所谓基本 HDLC帧，就是验证包类型是否由某个帧负责传送的帧。

链接控制协议(Link Control Protocol,LCP)，位于HDLC的顶部，用于协商数据链接选项，比如“最大接收单元”（MRU）等。最大传输单元表示链接方同意接收的数据报的最大字节数。

在配置PPP链接时，一个重要步骤是客户机验证。尽管它不是强制实施的，但对拨号线路来说，却是必不可少的。通常，被呼叫方（即服务器）要求客户机证明它知道密钥，并藉此对客户机的身份进行验证。如果呼叫方不能提供正确的密钥，连接就会中断。利用 PPP时，验证是双向的：也就是说，呼叫方也可以要求服务器验明它自己的身份。这两个验证过程彼此并不相干。不同类型的身份验证，采用的协议是不同的，这一点，我们将在后续小节进行讨论。一个叫做“密码验证协议”（Password Authentication,PAP），另一个叫做“盘问握手验证协议”（Challenge Handshake Authentication Protocol,CHAP）。

对通过数据链路路由的每个网络协议（比如IP和Apple Talk）来说，我们都可以利用相应的网络控制协议来对它们进行动态配置。例如，在打算通过链路发送IP数据报时，两端的PPP必须先对彼此使用的IP地址进行协商。这时所用的控制协议就是IPCP，即“Internet协议控制协议”。

除了通过链路发送标准的IP数据报外，PPP还支持IP数据报的Van Jacobson报头压缩。这种压缩技术将TCP包的报头缩小到3个字节。另外，它还用于CSLIP，即人们常说的VJ报头压缩。是否使用这种压缩，同样要在启动时，通过 IPCP进行协商。

7.1　PPP打开

在Linux中，PPP的作用分为两大类：其一，位于内核的低级 HDLC驱动程序；其二，处理不同控制协议的用户空间 pppd daemon。编写本书时，PPP的当前版本是Linux-ppp-1.0.0，其中包含内核PPP模块pppd和一个程序，该程序名为chat，用于拨号远程系统。

注意　当前PPP协议支持属于内核专用，诸如 pppd daemon之类的支持程序位于metalab.unc.edu/Pub/Linux/system/network/serial/ppp/（当前版本是ppp-2.3.4）。

PPP内核驱动程序是迈克·克拉翰编写的。 pppd衍生于一个免费的PPP执行程序，适用于Sun和386BSD机器，是德鲁·帕金斯和其他人一起编写的。由艾尔·依伊尔移植到 Linux中。

像SLIP一样，PPP是通过一种特殊的线路规则来实现的。要把串行线路当作 PPP链接使用，应该像以往一样，先通过 Modem建立一条链接，再把该线路转换为 PPP模式。这种模式下，收到的所有数据都被传递到 PPP驱动程序，驱动程序再对收到的这些 HDLC帧进行有效性检查（每个HDLC帧都带有一个16位的校验和），然后再解开封装，并开始分发数据。目前，可以选择性地采用Van Jacobson报头压缩，对IP数据报进行处理。如果有 IPX支持，PPP驱动程序还能够对IPX包进行处理。

内核驱动程序在 pppd的帮助下，得到了增强。在链接发送真正的网络通信之前，PPP daemon会执行必要的初始化和身份验证阶段。可通过许多选项，对 pppd的行为进行调整。由于PPP非常复杂，要想在一章里，全面介绍它是不可能的。因此，本书不能涵括 pppd的方方面面，只能为大家做一个简要的介绍。关于 PPP的更多详情，可参考 PPP手册和pppd源文件中的 README，它们将有助于你深入了解本书中没有介绍到的内容。如果在参阅所有文档之后，还难解心中的困惑，建议访问 comp.protocols.ppp和linux.dev.ppp，这个地方群英汇萃，多是开发pppd的专家。

7.2 运行pppd

在打算通过PPP链接接入因特网时，必须设立一些基本的网络特性，比如 loopback设备和解析器。关于这两者的设立，我们已在前一章讨论过。要注意的是：串行链接上使用的是 DNS，参见前一章。

如何利用 pppd建立一条PPP连接呢？好了，假设我们又从 vlager着手。现在，已经拨号呼叫PPP服务器c3po，登录到ppp账号。c3po服务器已经启动了自己的 PPP驱动程序。退出用于拨号的通信程序之后，执行下面的命令

```
## ppppd /dev/cua3 38400 crtstcs defaultroute
```

串行线路cua3便转换为PPP模式，并建立了一条通向 c3po服务器的IP链接。串行端口上所用的传输速率将是38400bps。crtscts选项打开了端口上的硬件设备握手，其绝对传输速率必须在9600bps以上。

启动之后，pppd所做的第一件事是利用 LCP与远程用户端协商一些链接特性。通常情况下，pppd试图协商的默认选项集都有用，所以我们不打算在这里深入讨论。关于 LCP的详情，我们将在后续小节讨论。

然后，pppd利用IPCP（IP控制协议），与其对等体一起，对 IP参数进行协商。由于我们没有指定任何特定的 IP地址，所以pppd将试图利用它获得的地址，这个地址是令解析器查找主机名所获得的。之后，通信双方向彼此通告各自的地址。

一般说来，这些默认设置都不会有错。即使你的机器运行于以太网，也可以在以太网和 PPP接口上采用同样的IP地址。不管怎么说，pppd都允许采用不同的地址，甚至要求你的对等体也采用某个特定的地址。我们将在下一小节讨论这些选项。

走完IPCP设立过程后，pppd将进行主机网络层的准备工作，以使用这条 PPP链接。首先，对已激活的第一条PPP链接，采用ppp0，第二条链接采用 ppp1，以此类推，把PPP网络接口配置为“点到点链接”。接下来，设立一条路由表条目，使之指向链接另一端的主机。在上面的示例中，pppd令默认路由指向c3po，因为我们把“默认路由”选项给了它（如果目前还没有路由的话，只能安装默认网络路由）。这样，将导致所有发向非本地网络上主机的数据报被发

送到c3po。pppd支持大量不同的路由选择方案，我们将在本章稍后详情讨论。

7.3 使用选项文件

pppd解析自己的命令行参数之前，会对一些文件进行查看，找出默认选项。这些文件中可能存在任何一个合法的命令行参数，它们分散在不同的命令行中。 hash符号引入了译注。

第一个选项文件是/etc/ppp/options，该文件通常是pppd启动时的第一个查看对象。利用它来设置全局默认选项倒是个好主意，因为它允许你防止你的用户进行那些破坏网络安全性的操作。例如，如果想要 pppd要求某种形式的身份验证（非 PAP，即CHAP），就要在该文件中增添auth选项。网络用户是不能修改这个选项的，所以要和不在身份验证数据库内的任何一个系统建立PPP链接是完全不可能的。

继/etc/ppp/options之后，pppd查看的第二个选项文件是用户根目录中的 .pprc文件。它允许每个用户指定他／她自己的默认选项集。

/etc/ppp/options选项文件示例：

```
# Global options for pppd running on vlager.vbrew.com
auth                # require authentication
userhostname        # use local hostname for CHAP
lock                # use UUCP -style device locking
domain vbrew.com    # our domain name
```

前两类选项文件适用于身份验证，我们将在稍后详细讨论。 lock关键字令pppd采用标准的UUCP设备锁定方法。有了这一约定，访问串行设备（比如 /dev/cua3）的每个进程都会在UUCP假脱机目录中，创建一个名为 LCK...cua3的锁定文件，表示该设备正在使用中。对 PPP正在使用的串行设备来说，这样可以有效地防止其他程序（比如 minicom和uucico）也来打开这个串行设备。

全局配置文件提供这些选项的原因在于：禁止网络用户对上面所说的选项进行修改，为整个网络的安全提供一定的保障。但请注意，有些选项是可以修改的，比如连接字串。

7.4 用chat拨出

上一个示例中，在启动 pppd之前，必须手工建立连接，这样带来的不便，可能是很多人都不愿意经历的。和 dip不一样， pppd没有可以用于拨叫远程系统和登录的脚本编写语言，但它也不纯粹依赖于某些外部程序或外壳脚本。利用连接命令行选项，便可把执行脚本编写的命令交给pppd。然后， pppd将该命令的标准输入和输出重新定向到串行线路。有用的编写语言之一是Expect，它是唐·利贝斯编写的。它是一种基于 Tcl的功能非常强劲的语言，简直就是为这类应用量身定做的。

pppd封装内有一个类似的程序，名为 chat（对话），它允许你指定一个 UUCP样式的对话脚本。对话脚本基本上由一个交替式的字串序列组成，这些字串是我们希望从远程系统中收到的（即expect字串），以及我们准备发送的应答。下面，我们将分别调用希望收到的和准备发送的字串。这是从一个对话脚本中摘录出来的典型字串：

```
ogin: b1ff ssword: s3krst
```

它要求对话等待远程系统发送登录提示并返回登录名 b1ff。我们只有等待登录：登录提示

是大写L，还是小写l，都无关紧要。下一个字串是 expect字串，它也会令对话等待密码提示，并发送应答中的密码。

这基本上就是一个对话脚本所包含的内容。当然，拨号到 PPP服务器的一个完整脚本中，还必须包含恰当的Modem命令。假如你的Modem能够识别贺氏标准的指令集，而且服务器的电话号码是318714，那么，与c3po建立连接的完整对话脚本调用如下所示：

```
$ chat -v ' ATZ OK ATDT318714 CONNECT ' ogin: ppp word: GaGariN
```

根据定义，第一个字串必须是一个 expect字串，但由于modem在我们剔出这个字串之前，是没有任何反应的，所以在指定一个空字串之后，我们就将对话跳到第一个 expect字串。然后，发送ATZ命令，这是采用贺氏标准的 modem所用的“ 重新设定 ”命令，并等待 modem应答（OK）。下一个字串向对话发送 dial命令的同时，还发送了电话号码，并希望在应答中收到CONNECT消息。随后又是一个空字串，因为我们不打算发送任何命令，只是等待登录提示。对话脚本的其余部分的工作原理和上面描述的完全一样。

-v选项会令对话把所有的活动记录到系统日志 daemon的loacal2设备中。（如果对syslog.conf进行编辑，将这些日志消息重定向到一个文件中，一定要确保该文件是非全球可读型，因为默认情况下，该对话也会对整个对话脚本进行记录 —— 包括密码和所有字串）。

在命令行指定对话脚本是很不安全的，因为网络用户用 ps命令，就能够看到一个进程的命令行。怎样避免呢？把对话脚本放在一个诸如 dial-c3po的文件内。令对话从该文件中读取脚本，而不是为它指定后面跟该文件名的 -f选项，以从命令行读取。完整的 pppd结果如下所示：

```
# pppd connect “ chat -f dial-c3po ” /dev/cua3 38400 -detach \crtscts modem
defaultroute
```

除了指定拨号脚本的连接选项外，我们还在命令行增加了两个选项： nodetach（该选项吩咐pppd不要脱离控制台，转为后台进程）和 modem关键字。modem关键字令pppd在串行设备上执行一些modem专有的动作，比如在拨号前后，挂断线路。如果不用这个关键字， pppd就不会对端口的DCD线路进行监控，从而导致远程终端意外挂断时， pppd不会对该线路进行保护。

上面的示例相当简单；对话可以接受更为复杂的对话脚本。其中一个最有用的特性是能够指定终止对话的错误字串。典型的终止字串是诸如 BUSY、NO CARRIER之类的消息，被呼叫的号码正忙或不能接受电话接入时， modem常常会产生此类消息。为了使对话能立即辨认出这些消息，而不是等待超时，可以在脚本开始处，利用 ABORT关键字指定这些消息：

```
$ chat -v ABORT BUSY ABORT ' NO CARRIER' " ATZ OK ...
```

也可采用类似的方式，插入 TIMEOUT选项，修改对话脚本各部分的超时值。有关详情，请查看对话手册。

有时，大家可能还想看到对话脚本各部分的违例情况。比如，在没有收到远程端的登录提示时，大家可能想发送一个 BREAK，或一个回车。这是通过在 expect字串后添加一个子脚本来完成的。子脚本由一序列 send-和expect-字串组成，与一个完整的脚本一样，用连字号 -隔开。只要规定时间内没有收到希望收到的字串，就会开始执行子脚本。在上面的示例中，我们将对对话脚本作出以下的修改：

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

现在，当对话看不到远程系统发送的登录提示时，就会通过先发送一个 BREAK的方式，执行子脚本，然后再次等待登录提示。如果出现提示，脚本就会一如既往地运行，否则，就会因出现错误而中断。

7.5 PPP设置的调试

默认情况下，pppd会将所有的警告和错误消息记录在系统日志的 daemon设备中。所以，你必须在syslog.conf中添加一个条目，使其重定向到一个文件中，甚至可以定向到控制台，不然，系统日志就会将这些消息丢弃。下面的条目将所有警告和错误消息统统发到 /var/log/ppp-log：

```
daemon.*     /var/log/ppp-log
```

如果你的PPP设置不能立即运行，那么查看这个日志文件，将提示你错在哪里。如果仍然没有解决问题，还可利用 debug选项，打开特别调试输出。它将使 pppd把所有收发控制包的内容统统记录在系统日志内。这样一来，所有消息都能进入 daemon设备。

最后，PPP最突出的特性便是启用内核级的调试。这是通过调用带有 kdebug选项的pppd来完成的。它后面跟着一个数字化参数，该参数是下列值的按位“ 或 ”：1)代表一般调试消息；2)代表打印所有接入 HDLC帧的内容；3)是令驱动程序打印所有流出 HDLC帧的内容。为捕捉内核调试消息，必须运行对 /proc/kmsg文件进行读取的 syslogd daemon，或者klogd daemon。两者都将内核调试定向到系统日志的内核设备。

7.6 IP配置选项

IPCP用于协商链路配置时的两个 IP参数。通常情况下，每个对等体都可发送一个“ IPCP配置请求 ”包，表示它想修改哪些默认值，并且将它们改为什么值。远程端收到之后，便依次检查各个选项，然后，要么接受它，要么否决它。

pppd将对大量的IPCP选项进行协商。要对这一协商进行调整，可通过不同的命令行选项来完成，如下所示。

7.6.1 IP地址的选择

上面的示例中，我们让 pppd拨号c3po，并建立了一条IP链接，没有预计在链接的任何一端选择特定的IP地址。相反地，我们选择 vlager的地址来作为本地IP地址,而让c3po仍然用它自己的地址。但是，有时，能够对链接两端的地址进行控制是非常有用的。 pppd支持IP地址的变更。

要想得到特定的IP地址，一般需要提供带有下一选项的 pppd：

```
local addr:remote addr
```

local_addr和remote_addr必须指定为点分四段式或主机名（这个选项中采用主机名会对CHAP身份验证产生影响。详情参考下面的 CHAP小节）。这样，pppd就会试图将第一个地址作为自己的IP地址，第二个地址作为其对等体的 IP地址。如果IPCP协商过程中，其对等体否决了这两个地址，就不能建立 IP链接了（可通过为 pppd提供ipcp-accept-local和ipcp-accept-remote这两个远程选项，令其对等体 PPP忽略前面提供的IP地址。有关详情，可参考手稿）。

如果只想设立本地 IP地址，接受对等体使用的任何一个地址，省略 remote_addr即可。例

如，要想vlager不采用它自己的IP地址，而使用地址130.83.4.27，就应在命令行上为它提供130.83.4.27:。类似地，如果只想设立远程地址，可令local_addr为空。默认情况下，pppd将采用与你的主机名相关的那个地址。

对那些控制多个客户站点的PPP服务器来说，它们通常动态分配地址：只对来访的系统分配地址，也就是说，登录主机所分配到的地址是动态的。在拨叫这类服务器时，必须保证pppd不从服务器发出任何特殊IP地址请求，但可以接受服务器要求你使用的那个地址。这意味着禁止你指定local_addr参数。另外，对你来说，必须采用noipdefault选项，该选项将令pppd等待对等体提供IP地址，而不是采用本地主机的地址。

7.6.2 通过PPP链路的路由

设置网络接口之后，pppd通常只设置一条通向其对等体的主机路由。如果远程主机在局域网上，你应该确信还能够连接到对等体以后的那些主机：也就是说，还必须设置一条网络路由。

由上可知，利用defaultroute（默认路由）选项，可以要求pppd设置默认路由。如果你拨叫的PPP服务器担当你的Internet网关这一角色，这个选项就会变得相当有用。

另一方面，如果你的系统只是一个单一主机的网关，情况也会非常容易处理。比如，以Virtual Brewery的一名员工为例，他的主机名为Loner。在通过PPP链路连接到vlager时，他使用的是Brewery子网上的一个地址。在vlager这一端，我们在pppd后添加了proxyarp选项，它将为Loner主机安装一个代理ARP条目。如此一来，我们就可以从Brewery和Winery的任何一台主机，访问Loner主机。

但是，事情并不如想像中的那样简单。以链接两个局域网为例，通常要求增加一条特殊的网络路由，因为这类网络都可能有自己的默认路由。另外，让链接的对等体都将PPP链接用作默认路由，将产生一个循环，也就是说不知其目的地的数据包将在对等体间不停地ping、pong，直到这些包的生存期满为止。

在将子以太网连接到vlager时，它将一如既往地把默认路由指向vlager。但在vlager这一端，我们必须为通过子以太网的子网3安装一条网络路由。鉴于此，我们采用了一个以前尚未谈到的pppd特性——ip-up命令。这是一个外壳脚本或程序，位于/etc/ppp（在配置完PPP接口之后执行的）。ip-up命令是随下面的参数一起调用的：

```
ip-up iface device speed local addr remote addr
```

iface命名网络接口使用的名字，device是串行设备文件所用的路径名（如果是stdin/stdout，路径名则是/dev/tty），speed是设备的传输速率。local_addr和remote_addr以点分四段式给出链路两端的IP地址。我们的示例中，ip-up脚本中可能包含下面的代码片段：

```
#!/bin/sh
case $5 in
191.72.3.1)             # this is sub-etha
        route add -net 191.72.3.0 gw 191.72.3.1;;
esac
exit 0
```

类似地，在PPP链路再次取消之后，利用/etc/ppp/ip-down来撤消ip-up的所有动作。

但是，路由并没有因此而结束。我们在两个PPP主机上都设置了路由表，但迄今为止，这

两个网络上的其他主机对PPP链路还一无所知。如果处于次要地位的所有主机都有各自指向子以太网的默认路由，而且所有Brewery主机都可通过默认路由到达vlager，其他主机知不知道PPP链路都无关紧要。但如果不是上面提到的情况，通常只有选用 gated之类的路由daemon。在vlager主机上创建网络路由之后，路由 daemon就会向邻近子网上的所有主机广播这条新的网络路由。

7.7 链路控制选项

由上可知，我们已见识过LCP，它用于协商链接特性和测试链路。

LCP协商的最重要的两个选项是“最大接收单元”和“异步控制字符映射”。此外还有许多LCP配置选项，但和我们这里讨论的主题不太相干。有关详情参见 RFC-1548。

异步控制字符映射，一般称为async map，用于电话线之类的异步链路上，用来标识必须避开的控制字符（用一个特殊的两字符序列来替换它）。例如，有些配置不当的modem可能在收到XOFF后，就会阻塞，这种情况下，人们可能会想避开用于软件握手的 XON和XOFF字符。其他的还有Ctr+]（Telnet必须避开的字符）。在asyn map中指定从0到32的带有ASCII代码的所有字符后，PPP就能避开它们了。

asyn map是一个32位宽的位图，带有对应ASCII NUL字符的最无意义的位，和对应ASCII 31的最有意义的位。如果设置了一位，就表示在通过链路发送该位之前，必须避开它的对应字符。最初，asyn map被设置为0xffffffff，也就是说，将避开所有的控制字符。

要想告诉你的对等体，不必避开全部控制字符，只避开少数几个时，可利用 asyncmap选项，为pppd指定一个新的asyncmap。例如，如果必须避开的字符只有^S和^Q（ASCII 17和19，常用于XON和XOFF），采用下面的选项即可：

```
asyncmap 0x000A0000
```

“最大接收单元”（简称MRU），通知对等体我们希望收到的HDLC帧的最大字节数。尽管它令人想起MTU值（最大传输单元），但两者几乎没有共同之处。 MTU是内核连网设备参数，说明的是接口能够处理的帧的最大字节数。而 MRU更偏向于建议远程端不要生成大于 MRU值的帧；但接口必须能够接收的帧不得超过 1 500字节。

因此，对链路能够传输的帧选择一个 MRU是非常简单的，重要的是选择最适合自己的流通量。如果试图在链路上运行交互式应用程序，最好把 MRU设置为低于296字节的值，如此一来，在偶尔碰到一两个稍大一点的包（这里指的是 FTP会话）时，你的光标不会因此而“跳动不停”。要告诉pppd请求一个296字节的MRU值，就应该为它指定mru 296这个选项。但是，只有在没有取消VJ报头压缩时，小型MRU才有意义。

pppd也可识别两个LCP选项，这两个选项配置了协商进程的总体行为，比如链路中断之前，可以交换的最大配置请求数。除非你知道自己正在做什么，否则不应该改动它们。

最后，还有两个适用于LCP响应消息的选项。PPP定义了两条消息：响应请求和响应应答。pppd利用这一特性，检查一条链接是否仍处于运作过程中。可以利用 lcp-echo-interval选项和一个以秒计时的时间值来启用这一特性。如果在指定时间内，没有收到发自远程主机的帧，pppd会生成一个“响应请求”，并希望其对等体返回一个“响应应答”。如果其对等体没有作出应答，在发送特定数量的请求之后，这条链路就会中断。可利用 lcp-echo-failure选项，设置发送请求量。默认情况下，这个特性也是取消了的。

7.8 常规安全问题

一个配置不当的PPP daemon可能严重破坏你的网络。比如任何人都可把他们的机器插入你的以太网(这是最糟糕的)。本小节将讨论保证 PPP配置安全可靠的几个标准。

pppd存在的一个问题是配置网络设备和路由信息表时，要求拥有根特权。一般说来，这个问题的解决办法通常是运行 setuid root。

pppd允许用户设置大量不同安全方面的选项。为避免用户在计算这些选项之后，再启用它们，对你的网络进行恶意攻击，在此建议设置 global/etc/ppp/options文件（类似于示例文件）中的两个默认值。用户不能修改其中某些选项，比如身份验证选项。所以，从某种程度上来讲，也能提供一些安全保障。

当然，还必须对和你进行 PPP链接的系统进行保护。比如拥有对等体的某些身份验等等。另外，还不应该允许国外主机利用他们自己选择的 IP地址，如果允许的话，也应该为数不多。下一小节将就此详细讨论。

7.9 PPP身份验证

7.9.1 CHAP和PAP

利用PPP链路时，每个系统可能都要求其对等体采用下面两个协议之一来验明自己的身份。它们是：密码验证协议（ PAP）和盘问握手验证协议（ CHAP）。建立链接之时，链路两端都要求另一方验明各自的身份，无论是呼叫方，还是被呼叫方，都如此。在区别身份验证系统和身份验证者之前，先为大家谈谈“ 客户机 ”和“ 服务器 ”。对PPP daemon来说，通过发送标识需要哪个身份验证协议的另一个 LCP配置请求，便可要求其对等体进行身份验证。

PAP的运作基本上类似于普通登录过程。客户机向服务器发送用户名和一个（可以加密）密码，向服务器验明自己的身份，服务器将收到的用户名和密码和自己的机密数据库中保存的内容进行比较。这一技术有很大的漏洞，如果有人想获得密码的话，在该串行线路实行监听，就可以重复尝试对网络发起恶意攻击。

CHAP则不存在这些缺陷。使用 CHAP，身份验证者（也就是服务器）把自己的主机名和一个随机生成的“ 盘问 ”字串发送给客户机。客户机利用这个主机名查找相应的密钥，并把找到的密钥和盘问组合起来，用一个单向的散列函数对该盘问字串进行加密。其结果再随客户机的主机名一起，返回服务器。服务器再执行同样的计算，如果结果相同，就认可这个客户机。

pppd将CHAP和PAP所用的密钥保存在两个独立的文件内，分别是 /etc/ppp/chap-secrets和pap-secrets。在任何一个文件内输入一个远程主机名，均可很好地对用于自己与对等体之间的身份验证中的CHAP或PAP进行控制，反之亦然。

默认情况下， pppd不要求远程主机验明身份，但在远程主机发出验证请求时，它会同意验明自己的身份。由于 CHAP比PAP更为强壮，只要可能， pppd就会先尝试使用前一个协议。如果对等体不支持，或 pppd不能在自己的 chap-secrets文件内找到远程系统的 CHAP密钥，pppd就会转而使用PAP。如果同样没有其对等体的 PAP密钥，它就会拒绝进行身份验证。其结果是连接中断。

有几种方式可对上面的行为进行修改。例如，在收到 auth关键字时， pppd将要求其对等

体验明自身的身份。只要pppd的CHAP或PAP数据库内分别包含这个对等体的密钥，它就会同意使用CHAP或PAP。另外，还有别的选项，用于打开或关闭特殊身份验证协议，但在此不作讨论。详情参阅pppd手稿。

如果和你进行PPP链接的所有系统都同意验明自己的身份，就应该把auth选项放入global/etc/ppp/options文件内，并在chap-secrets文件中为每个系统定义密码。如果某个系统不支持CHAP，则在pap-secrets文件内为它增加一个密钥条目。通过这种方式，就可以保证没有通过身份验证的系统不能链接你的主机。

下面两个小节讨论两个PPP密钥文件：pap-secrets和chap-secrets。它们位于/etc/ppp下，其中包含许多三个（客户机、服务器和密码）一组的条目，有的后面还跟有IP地址清单。CHAP和PAP对客户机与服务器的解释是有区别的，与我们是向对等体验明身份，还是要求服务器向我们验明它们的身份有关。

7.9.2 CHAP密钥文件

在必须利用CHAP向某一服务器验明自身时，pppd将在pap-secrets文件内搜索带有客户机字段和服务器字段的条目。这里的客户机字段和服务器字段分别等同于CHAP盘问中的本地主机名和远程主机名。在要求对等体验明自身时，只有角色发生了变化：pppd查找带有客户机字段和服务器字段的条目，它们分别等同于客户机发出的CHAP响应中的远程主机名和本地主机名（注意，前后顺序发生了变化）。

下面是vlager的chap-secrets文件（双引号不属于密码，只是保护密码内的空格）：

```
# CHAP secrets for vlager.vbrew.com
#
# client          server            secret                addrs
#-----------------------------------
vlager.vbrew.com  c3po.lucas.com    "Use The Source Luke" vlager.vbr
c3po.lucas.com    vlager.vbrew.com  "riverrun, pasteve"   c3po.lucas
*                 vlager.vbrew.com  "VeryStupidPassword"  pub.vbrew.
```

在与c3po建立PPP链接时，c3po要求vlager利用CHAP，通过发送一个CHAP盘问的方式，验明自己的身份。然后，pppd在chap-secrets文件中查找客户机字段是vlager.vbrew.com，服务器字段是c3po.lucas.com（这个主机名选自CHAP盘问）的条目，并找到了上面列出的第一行。然后，pppd生成取自盘问字串和密钥（UseThe Source Luke）的CHAP应答，并把它发回c3po。

与此同时，pppd为c3po写一个CHAP盘问，其中包含一个独一无二的盘问字串及其完整资格主机名vlager.vbrew.com。c3po以刚才我们介绍的方式，构建了一个CHAP应答，并将该应答返回vlager。现在，pppd从应答中取出客户机主机名（c3po.vbrew.com），并在chap-secrets文件内搜索客户机为c3po，服务器为vlager的条目行。结果找到了第二行，所以pppd便把CHAP盘问和密钥riverrun与pasteve组合起来，并对它们进行加密，与c3po的CHAP应答中的结果进行比较。

仅供选择的第四个字段列出了一系列IP地址，这些地址都是第一个字段内的客户机能够接受的。它们可采用点分四段式，也可以是利用解析程序查找到的主机名。例如，IPCP协商过程中，如果c3po请求使用这个地址清单中没有的IP地址，请求就会遭到拒绝，而且IPCP也

会被关闭。在上面的示例文件中，限定 c3po只能使用自己的IP地址。如果地址字段为空，意味着所有的地址都不会遭到否决；“ - ”值可避免随客户机一起使用IP地址。

上面的chap-secrets示例文件中，第三行允许任何一台主机与 vlager建立PPP链接，因为客户机和服务器字段均为“ * ”（通配符）。唯一的要求是它们已知密钥并采用 pub.vbrew.com这一地址。带有通配主机名的条目可能出现在密钥文件中的任何地方，因为 pppd总是采用应用了服务器／客户机对的最具体的条目。

由上可知，远程主机名一直是由对等体在其发出的 CHAP盘问或应答包中提供的。默认情况下，本地主机名是调用 gethostname(2)函数产生的。如果已经把系统名设为你自己的未合格主机名，就必须另行利用domain（域）选项，为pppd提供域名。domain选项如下所示：

```
# pppd ...domain vbrew.com
```

这样， Brewery的域名便会增加到vlager，进行全部身份验证活动。修改 progpppd本地主机名的其他选项是 usehostname和name。在命令行上利用 local:varremote，指定本地IP地址，而且本地指的是本地主机名而不是点分四段表达式时， pppd就会把它用作本地主机名。有关详情，请参考pppd手册。

7.9.3 PAP密钥文件

PAP密钥文件的用法类似于CHAP文件。前两个字段总是包含用户名和服务器名；第三个字段保存的是PAP密钥。远程主机发出一个身份验证请求时， pppd便采用其服务器字段和用户名字段分别等同于请求中的本地主机名和用户名的条目。向其对等体验明自身后， pppd就从准备发送的行（其中的用户名字段和服务器字段分别等同于本地用户名和远程主机名）内选出密钥。

PAP密钥文件示例：

```
# /etc/ppp/pap-secrets
#
# user          server          secret          addrs
vlager-pap      c3po            cresspah1       vlager.vbrew.com
c3po            vlager          DonaldGNUth     c3po.lucas.com
```

第一行用于在与 c3po交流时，验明自己的身份。第二行描述了名为 c3po的用户是如何向我们验明它自己的身份的。

第一列中的vlager-pap名是我们发给c3po的用户名。默认情况下， pppd将把本地主机名用作其用户名，但也可通过user（用户）选项，后面跟上另一个主机名，另行指定。

从pap-secrets文件内选择用于向对等体验明身份的条目时， pppd必须知道远程主机名。由于它无法自行找出远程主机名，所以你必须在命令行上，利用 remotename（远程主机名）关键字指定它。例如，如果想采用上面的条目，向 c3po验明身份，必须在pppd的命令行增加下面的选项：

```
# pppd ...domain vbrew.com
```

第四个字段（以及随后的所有字段）中，可以像 CHAP密钥文件中那样，指定特定的主机能接受哪些IP地址。然后，对等体只需请求列出的地址。在示例文件中，我们要求 c3po使用它自己的真实IP地址。

注意，PAP是相当容易“受伤”的身份验证协议，所以我们建议大家尽可能使用 CHAP。这也是我们不打算深入讨论PAP的原因。如果想了解PAP的更多特性，可参考pppd手册。

7.10 PPP服务器的配置

把pppd当作服务器运行只是在命令行增加了恰当的选项而已。理想情况下，可创建一个特殊的账户，说一声ppp，并给它一个脚本或程序作为其登录外壳，外壳将利用这些选项调用pppd。例如，我们可在/etc/passwd内增加这一行：

ppp:*:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin

当然，大家可能还想采用不同于上面的uid和gid。这时别忘了用passwd命令，为上面的账户设置密码。

然后，ppplogin脚本就可能像下面这样：

```
#!/bin/sh
# ppplogin - script to fire up pppd on login
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

mesg命令禁止其他用户使用 write命令写入tty。stty命令关闭了字符响应。这是必要的，因为如果不这样，对等体发出的所有一切都会返回。上面给出的 pppd选项中，最重要的是-detach，因为它可防止pppd从控制tty中分离出去。如果我们不指定这个选项， pppd就会转入后台，令外壳脚本退出。这样将依次导致串行线路挂断，链接丢弃。 silent选项令pppd在开始发送包之前处于等待状态，直到它收到呼叫系统发来的一个包为止。这样可防止呼叫系统缓慢启动它的PPP客户机所产生的传输超时。 modem使pppd监控DTR线路，查看对等体是否已丢弃连接，crtscts则用于打开硬件握手。

除了上面提到的选项外，大家可能还想实施其他身份验证，比如在 pppd的命令行或全局选项文件内指定auth等等。如果想进一步了解打开和关闭个别的身份验证协议时所用的特定选项，可参考pppd手册。