Linux PPP HOWTO

Robert Hart, hartr@hedland.edu.au v2.2, 25 August 1996

This document shows how to connect your Linux PC to a PPP server, how to use PPP to link two LANs together and provides one method of setting up your Linux computer as a PPP server.

Contents

1 Introduction **4**

1.1 Clients and Servers 5

1.2 Differences between Linux distributions 5

2 IP Numbers **6**

3 Aim of this Document **8**

3.1 Setting up a PPP Client 8

3.1.1 Using DIP - don't, use CHAT instead 8

3.2 Setting up a PPP server 8

3.3 Linking two LANs or a LAN to the Internet using PPP 8

3.4 This document at present does NOT cover... 8

4 Software versions covered **9**

5 Other Useful/Important Documents **9**

5.1 Useful Linux Mailing Lists 10

6 Configuring your Linux Kernel **10**

6.1 Installing the Linux Kernel source 10

6.2 Knowing your hardware 11

6.3 Kernel compilation - the Linux 1.2.13 kernel 11

6.4 Kernel compilation - the Linux 1.3.x and 2.0.x kernels 12

6.5 Note on PPP-2.2 and /proc/net/dev 12

6.6 General kernel config considerations 13

7 Getting the Information you need about the PPP service **13**

7.1 Testing your Modem Connection for outgoing calls 14

8 A note about serial ports and speed capabilities **15**

9 Configuring your modem **15**

9.1 Note on Serial Flow Control 16

10 Using PPP and root privileges **17**

11 Setting up the PPP connection files **18**

11.1 The supplied options.tpl file 19

11.2 What options should I use? 25

11.2.1 /etc/ppp/options (NO PAP/CHAP) 25

11.2.2 /etc/ppp/options (using PAP/CHAP) 26

12 Setting up your /etc/resolv.conf file **26**

13 The PAP/CHAP secrets file **27**

13.1 The PAP secrets file 28

13.2 The CHAP secrets file 28

14 Setting up the PPP connection manually **29**

15 Automating your connections - Creating the connection scripts **32**

15.1 Connection scripts for Username/Password Authentication 32

15.2 The ppp-on script 33

15.3 Editing the supplied PPP startup scripts 34

15.3.1 The ppp-on script 34

15.3.2 Starting PPP at the server end 35

15.3.3 The ppp-on-dialer script 36

15.4 What a Chat script means... 36

15.5 A chat script for PAP/CHAP authenticated connections 37

15.6 The pppd debug and -f option_file options 38

16 Testing your connection script **38**

17 Shutting down the PPP link **41**

18 Debugging **41**

18.1 I compiled in PPP but Linux says I don't have it! . . . 42

18.2 I cannot set up a default route . . . 42

19 Linking two networks using PPP **43**

19.1 Setting up IP numbers . . . 43

19.2 Setting up the routing . . . 43

19.3 Network security . . . 44

20 After the link comes up... **44**

20.1 Special routing . . . 44

20.2 Handling email . . . 46

21 Shutting down the link **47**

22 Routing issues on a LAN **47**

22.1 Note on Security . . . 48

23 Getting Help when totally stuck **48**

24 Common Problems once the link is working **49**

24.1 I can't see beyond the PPP server I connect to . . . 49

24.2 I can send email, but not receive it . . . 50

24.3 Why can't people finger, WWW, gopher, talk etc to my machine? . . . 50

25 Using Internet services with Dynamic IP numbers **50**

25.1 Setting up email . . . 50

25.2 Setting Up a local Name server . . . 51

26 Setting up a PPP server **51**

26.1 Kernel compilation . . . 52

26.2 Overview of the server system . . . 52

26.3 Getting the software together . . . 53

26.4 Setting up standard (shell access) dialup. . . . 53

26.5 Setting up the PPP options files . . . 53

26.6 Setting pppd up to allow users to (successfully) run it . . . 54

26.7 Setting up the global alias for pppd 54

27 Using PPP across a null modem (direct serial) connection 55

Copyright

The copyright of this document is retained by the author. Permission is granted to distribute the document by electronic means and on CDs provided that it is kept entirely in its original format. Permission is also granted to print a copy of this document for personal use.

The republishing of this document in part or in whole without the permission of the copyright holder by any means other than as noted above is prohibited.

Distribution

This document will be posted to comp.os.linux.answers as new versions of the document are produced.It is also available in HTML format at:-

- *Linux Howto Index* `<http://sunsite.unc.edu/mdw/linux.html#howto>`

Other formats (SGML, ASCII, postscript, DVI) are available from *Howtos - other formats* `<ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/other-formats>`

As sunsite.unc.edu carries a very heavy load, please use an appropriate mirror site closest to you.

Acknowledgements

A growing number of people have provided me with assistance in preparing this document. Special thanks go to Al Longyear for the guidance on PPP itself (if there are mistakes here, they are mine not his), Greg Hankins (maintainer of the Linux Howto system) and Daniel Berinson for assistance with the linuxdoc-sgml package when it started to choke on my text (3 days before a publication dead line, no less!) and Debi Tackett (of MaximumAccess.com) for many helpful suggestions on style, content order, logic and clarity of explanations.

Finally, to the many people who have contacted me by email offering comments - my thanks. As with all HOWTO authors, the satisfaction of helping is all the payment we receive and it is enough. By writing this HOWTO I am repaying in a small way the debt I - and all other Linux users - owe to the people who write and maintain our OS of choice.

1 Introduction

PPP (the Point to Point Protocol) is a mechanism for creating and running IP (the Internet Protocol) and other network protocols over a serial link - be that a direct serial connection (using a null-modem cable), over a telnet established link or a link made using modems and telephone lines. Using PPP, you can connect your Linux PC to a PPP server and access the resources of the network to which the server is connected (almost) as if you were directly connected to that network.

You can also set up your Linux PC as a PPP server, so that other computers can dial into your computer and access the resources on your local PC and/or network.

As PPP is a peer-to-peer system, you can also use PPP on two Linux PCs to link together two networks (or a local network to the Internet).

One major difference between PPP and an Ethernet connection is of course speed - a standard Ethernet connection operates at 10 Mbs (million bits per second) maximum theoretical throughput, whereas a modem operates at speeds up to 33.6 kbps (thousand bits per second). Also, depending on the type of PPP connection, there may be some limitations in usage of some applications and services.

1.1 Clients and Servers

PPP is strictly a peer to peer protocol; there is (technically) no difference between the machine that dials in and the machine that is dialed into. However, for clarity's sake, it is useful to think in terms of **servers** and **clients**.

When you dial into a site to establish a PPP connection, you are a **client**. The machine to which you connect is the **server**.

When you are setting up a Linux box to receive and handle dial in PPP connections, you are setting up a PPP **server**.

Any Linux PC can be both a PPP server and client - even simultaneously if you have more than one serial port (and modem if necessary). As stated above, there is no real difference between clients and servers as far as PPP is concerned, once the connection is made.

This document refers to the machine that initiates the call (that "dials in") as the **CLIENT**, whilst the machine that answers the telephone, checks the authentication of the dial in request (using user ids, passwords and possibly other mechanisms) is referred to as the **SERVER**.

The use of PPP as a client to link one or more machines at a location into the Internet is probably, the one in which most people are interested - that is using their PC as a client.

The procedure described in this document will allow you to establish and automate your Internet connection.

This document will also give you guidance in setting up your Linux PC as a PPP **server** and in linking two LANs together (with full routing) using PPP (this is frequently characterised as establishing a WAN - wide area network - link).

1.2 Differences between Linux distributions

There are many different Linux distributions and they all have their own idiosyncrasies and ways of doing things.

In particular, there are two different ways a Linux (and Unix) computer actually starts up, configures its interfaces and so forth.

These are **BSD style system initialisation** and **System V system initialisation**. If you dip into some of the Unix news groups, you will find occasional religious wars between proponents of these two systems. If that sort of thing amuses you, have fun burning bandwidth and join in!

Possibly the two most widely used distributions are

- Slackware
  which uses BSD style system initialisation
- Red Hat (and its sibling Caldera)
  which uses SysV system initialisation

BSD style initialisation keeps its initialisation files in `/etc/...` and these files are:-

```
/etc/rc
/etc/rc.local
/etc/rc.serial
```

System V initialisation keeps its initialisation files in `/etc/rc.d/...` and a number of subdirectories under there:-

```
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 init.d
-rwxr-xr-x   1 root     root         1776 Feb  9 05:01 rc
-rwxr-xr-x   1 root     root          820 Jan  2  1996 rc.local
-rwxr-xr-x   1 root     root         2567 Jul  5 20:30 rc.sysinit
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc0.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc1.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc2.d
drwxr-xr-x   2 root     root         1024 Jul 18 18:07 rc3.d
drwxr-xr-x   2 root     root         1024 May 27  1995 rc4.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc5.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc6.d
```

If you are trying to track down where your Ethernet interface and associated network routes are actually configured, you will need to track through these files to actually find where the actual commands are that do this.

On some installations (for example Red Hat and Caldera), there is a X Windows configured PPP dial up system. This HOWTO does not cover these distribution specific tools. If you are having problems with them, contact the distributors directly!

2 IP Numbers

Every device that connects to the Internet must have its own, unique IP number. These are assigned centrally by a designated authority for each country.

If you are connecting a local area network (LAN) to the Internet, **YOU MUST** use an IP number from your own assigned network range for all the computers and devices you have on your LAN. You **MUST**

NOT pick IP numbers out of the air and use these whilst connecting to another LAN (let alone the Internet). At worst this will simply not work at all and could cause total havoc as your 'stolen' IP number starts interfering with the communications of another computer that is already using the IP number you have picked out of the air.

Please note that the IP numbers used throughout this document (with some exceptions) are from the 'unconnected network numbers' series that are reserved for use by networks that are not (ever) connected to the Internet.

There are IP numbers that are specifically dedicated to LANs that do not connect to the Internet. The IP number sequences are:-

- One A Class Address
  10.0.0.0 - 10.255.255.255
- 16 B Class Addresses
  172.16.0.0 - 172.31.255.255
- 256 C Class Addresses
  192.168.0.0 - 192.168.255.255

If you have a LAN for which you have **not** been allocated IP numbers by the responsible authority in your country, you should use one of the network numbers from the above sequences for your machines.

These numbers should **never** be used on the Internet. However, they can be used for the local Ethernet on a machine that is connecting to the Internet. This is because IP numbers are actually allocated to a network interface, not to a computer. So whilst your Ethernet interface may use 10.0.0.1 (for example), when you hook onto the Internet using PPP, your PPP interface will be given another (and valid) IP number by the server. Your PC will have Internet connectivity, but the other computers on your LAN will not.

However, using Linux and the IP masquerade capabilities of the ipfwadm software, you can connect the rest of your LAN to the Internet (with some restriction of services).

For more information on how to do this see the IP Masquerade mini-HOWTO at *Linux IP Masquerade mini HOWTO* `<http://sunsite.unc.edu/mdw/HOWTO/mini/IP-Masquerade>`

For most users, who are connecting a single machine to an Internet service provider via PPP, obtaining an IP number (or more accurately, a network number) will not be necessary.

If you wish to connect a small LAN to the Internet, many Internet Service Providers (ISPs) can provide you with a dedicated subnet (a specific sequence of IP numbers) from their existing IP address space.

For users, who are connecting a single PC to the Internet via an ISP, most providers use **dynamic** IP number assignment. That is, as part of the connection process, the PPP service you contact will tell your machine what IP number to use for the PPP interface during the current session.

With dynamic IP numbers, you are **not** given the same IP number each time you connect. This has implications for server type applications on your Linux machine such as sendmail, ftpd, httpd and so forth. The limitations of service due to dynamic IP number assignment (and ways to work around these, where possible) are discussed later in the document.

3 Aim of this Document

3.1 Setting up a PPP Client

This document provides guidance to people who wish to use Linux and PPP to dial into a PPP server and set up an IP connection using PPP. It assumes that PPP has been compiled and installed on your Linux machine (but does briefly cover reconfiguring/recompiling your kernel to include PPP support).

3.1.1 Using DIP - don't, use CHAT instead

Whilst DIP (the standard way of creating a SLIP connection) can be used to forge a PPP connection, DIP scripts are generally quite complex. For this reason, this document does NOT cover using DIP to forge a PPP connection.

Instead, this document describes the standard Linux PPP software (chat/pppd).

3.2 Setting up a PPP server

This document provides guidance on how to configure your Linux PC as a PPP server (allowing other people to dial into your Linux PC and establish a PPP connection).

You should note that there are a myriad of ways of setting up Linux as a PPP server. This document (currently) gives one method - that used by the author to set up several small PPP server (each of 16 modems).

This method is known to work well. However, it is not necessarily the best method. If other users have particularly clever PPP server setups, please feel free to email information on them to the author of this HOWTO.

3.3 Linking two LANs or a LAN to the Internet using PPP

This document provides (basic) information on linking two LANs or a LAN to the Internet using PPP.

3.4 This document at present does NOT cover...

- Connecting and configuring a modem to Linux (in detail)
  See the Serial-HOWTO
- Using DIP to make PPP connections
  Use chat instead...
- Using socks or IP Masquerade
  There are perfectly good documents already covering these two packages.

4 Software versions covered

This HOWTO assumes that you are using a Linux 1.2.x kernel with the PPP 2.1.2 software or Linux 1.3.X/2.0.x and PPP 2.2

It is possible to use PPP 2.2.0 with kernel 1.2.13. However, to do so requires kernel patches. This document does **NOT** cover this mix of software.

Also, you should particularly not that you cannot use the PPP 2.1.2 software with Linux kernel version 2.0.X.

Please note that this document does **NOT** cover problems arising from the use of loadable modules for Linux kernel 2.0.x. Please see the kerneld mini-HOWTO and the kernel/module 2.0.x documentation (in the Linux 2.0.x source tree at `/usr/src/linux/Documentation/...`).

As this document is designed to assist new users, it is highly recommended that you use a version of Linux and the appropriate PPP version that are known to be stable together.

5 Other Useful/Important Documents

Users are advised to read :-

- the documentation that comes with the PPP package
  (Look in `/usr/doc...`)
- the pppd and chat man pages
  (use `man chat` and `man pppd` to explore these)
- the Linux Network Administration Guide (NAG)
  The Network Administrators' Guide
  `<http://sunsite.unc.edu/mdw/LDP-books/nag-1.0/nag.html>`
- the Net-2/3 HOWTO
  Linux NET-2/3-HOWTO `<http://sunsite.unc.edu/mdw/HOWTO/NET-2-HOWTO.html>`
- Linux kernel documentation in /usr/src/linux/Documentation
- The excellent Unix/Linux books published by O'Reilly and Associates. (*O'Reilly and Associates On-Line Catalogue* `< http://www.ora.com/>`). If you are new to Unix/Linux, **run** (don't walk) to your nearest computer book shop and invest in a number of these immediately!

The best general starting point for Linux documentation is *The Linux Documentation Project Home Page* `<http://sunsite.unc.edu/mdw/>`

Whilst you can use this document to create your PPP link without reading any of these documents, you will have a far better understanding of what is going on if you do so! You will also be able to address problems yourself (or at least ask more intelligent questions on the comp.os.linux... newsgroups).

These documents (as well as various others, including the relevant RFCs) provide additional and more detailed explanation than is possible in this HOWTO.

If you are connecting a LAN to the Internet using PPP, you will need to know a reasonable amount about TCP/IP networking. In addition to the documents above, you will find the O'Reilly books "TCP/IP Network Administration" and "Building Internet Firewalls" of considerable benefit!

5.1 Useful Linux Mailing Lists

There are many Linux mailing lists that operate as a means of communication between users of many levels of ability. By all means subscribe to those that interest you and contribute your expertise and views.

A word to the wise: some lists are specifically aimed at "high powered" users and/or specific topics. Whilst no-one will complain if you 'lurk' (subscribe but don't post messages), you are likely to earn heated comments (if not outright flames) if you post 'newbie' questions to inappropriate lists! This is not because guru level users hate new users, but because these lists are there to handle the specific issues at particular levels of difficulty.

By all means join the lists that offer open subscription, but keep you comments relevant to the subject of the list!

A good starting point for Linux mailing lists is *Linux Mailing List Directory*
`<http://summer.snu.ac.kr/ djshin/linux/mail-list/index.shtml>`

6 Configuring your Linux Kernel

In order to use PPP, your Linux kernel must be compiled to include PPP support. Obtain the Linux source code for your kernel if you do not already have this - it belongs in `/usr/src/linux` on Linux's standard file system.

Check out this directory - many Linux distributions install the source tree (the files and subdirectories) as part of their installation process.

Linux kernel sources can be obtained by ftp from `sunsite.unc.edu` or its mirror sites.

6.1 Installing the Linux Kernel source

The following are brief instructions for obtaining and installing the Linux kernel sources. Full information can be obtained from *The Linux Kernel HOWTO*
`<http://sunsite.unc.edu/mdw/HOWTO/Kernel-HOWTO.html>`.

In order to install and compile the Linux kernel, you need to be logged in as root.

1. Change to the `/usr/src` directory
   `cd /usr/src`
2. Check in `/usr/src/linux` to see if you already have the sources installed.
3. If you don't have the sources, get them from *Linux kernel source directory*
   `<ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0>`

If you are looking for earlier versions of the kernel (such as 1.2.X), these are kept in *Old Linux kernel source directory* <ftp://sunsite.unc.edu/pub/Linux/kernel/old>.

4. Choose the appropriate kernel - usually the most recent one available is what you are looking for. Retrieve this and put the source tar file in `/usr/src`.
   Note: a 'tar' file is an archive - possibly compressed, as are the Linux kernel source tar files - containing many files in a number of directories. It is the Linux equivalent of a DOS zip file.

5. If you already have the Linux sources installed but are upgrading to a new kernel, you must remove the old sources. Use the command
   `rm -rf /usr/src/linux`

6. Now uncompress and extract the sources using the command
   `tar xzf linux-2.0.6.tar.gz`

7. Now, `cd /usr/src/linux` and read the README file. This contains an excellent explanation of how to go about configuring and compiling a new kernel. Read this file (it's a good idea to print it out and have a copy handy whilst you are compiling until you have done this enough times to know your way around).

6.2 Knowing your hardware

You MUST know what cards/devices you have inside your PC if you are going to recompile your kernel!!! For some devices (such as sound cards) you will also need to know various settings (such as IRQ's, I/O addresses and such).

6.3 Kernel compilation - the Linux 1.2.13 kernel

To start the configuration process, follow the instructions in the README file to properly install the sources. You start the kernel configuration process with

```
make config
```

In order to use PPP, you must configure the kernel to include PPP support (PPP requires BOTH pppd AND kernel support for PPP).

```
PPP (point-to-point) support (CONFIG_PPP) [n] y
```

Answer the other make config questions according to the hardware in your PC and the features of the Linux operating system you want. Then continue to follow the README to compile and install your new kernel.

The 1.2.13 kernel creates only 4 PPP devices. For multi- port serial cards, you will need to edit the kernel PPP sources to obtain more ports. (See the README.linux file that comes as part of the PPP-2.1.2 distribution for full details of the simple edits you need to make).

Note: the 1.2.13 configuration dialogue does NOT allow you to go backwards - so if you make a mistake in answering one of the questions in the `make config` dialogue, exit by typing CTRL C and start again.

6.4 Kernel compilation - the Linux 1.3.x and 2.0.x kernels

For Linux 2.0.x, you can use a similar process as for Linux 1.2.13. Again, follow the instructions in the README file to properly install the sources. You start the kernel configuration process with

```
make config
```

However, you also have the choice of

```
make menuconfig
```

This provides a menu based configuration system with online help that allows you to jump around in the configuration process.

There is also a highly recommended X windows based configuration interface

```
make xconfig
```

You can compile PPP support directly into your kernel or as a loadable module. If you only use PPP some of the time that your Linux machine is operating, then compiling PPP support as a loadable module is recommended. Using 'kerneld', your kernel will automatically load the module(s) required to provide PPP support when you start your PPP link process. This saves valuable memory space: no part of the kernel can be swapped out of memory, but loadable modules are automatically removed if they are not in use.

To do this, you need to enable loadable module support:-

```
Enable loadable module support (CONFIG_MODULES) [Y/n/?] y
```

To add PPP kernel support, answer the following question:-

```
PPP (point-to-point) support (CONFIG_PPP) [M/n/y/?]
```

For a PPP loadable module, answer **M**, otherwise for PPP compiled in as part of the kernel, answer **Y**.

Unlike kernel 1.2.13, kernel 2.0.x creates PPP devices on the fly as needed and it is not necessary to hack the sources to increase available PPP device numbers at all.

6.5 Note on PPP-2.2 and /proc/net/dev

If you are using PPP-2.2, you will find that a side effect of the 'on the fly' creation of the ppp devices is that no devices show up if you look in the `/proc/net` file system until a device is created by starting up pppd:-

```
[hartr@archenland hartr]$ cat /proc/net/dev
Inter-|   Receive                |  Transmit
 face |packets errs drop fifo frame|packets errs drop fifo colls carrier
    lo:  92792    0    0    0    0    92792    0    0    0     0    0
  eth0: 621737   13   13    0   23   501621    0    0    0  1309    0
```

Once you have one (or more) ppp services started, you will see entries such as this (from our ppp server):-

```
[root@kepler contrib]# cat /proc/net/dev
Inter-|   Receive                   |  Transmit
 face |packets errs drop fifo frame|packets errs drop fifo colls carrier
    lo: 428021    0    0    0    0   428021    0    0    0     0    0
  eth0:4788257  648  648  319  650  1423836    0    0    0  4623    5
  ppp0:   2103    3    3    0    0     2017    0    0    0     0    0
  ppp1:  10008    0    0    0    0     8782    0    0    0     0    0
  ppp2:    305    0    0    0    0      297    0    0    0     0    0
  ppp3:   6720    7    7    0    0     7498    0    0    0     0    0
  ppp4: 118231  725  725    0    0   117791    0    0    0     0    0
  ppp5:  38915    5    5    0    0    28309    0    0    0     0    0
```

6.6 General kernel config considerations

If you are setting up your Linux PC as a PPP server, you must compile in IP forwarding support. This is also necessary if you want to use Linux to link to LANs together or your LAN to the Internet.

If you are linking a LAN to the Internet (or linking together two LANs), you should be concerned about security. Adding support for IP firewalls to the kernel is probably a MUST!

You will also need this if you want to use IP masquerade to connect a LAN that uses any of the above mentioned 'unconnected' IP network numbers.

Once you have installed and rebooted your new kernel, you can start configuring and testing your PPP link(s).

7 Getting the Information you need about the PPP service

Before you can establish a PPP connection with a server, you need to obtain the following information (from the sysadmin/user support people of the PPP server):-

- The telephone number(s) to dial for the service
  If you are behind a PABX, you also need the PABX number that gives you an outside dial tone - this is frequently digit zero (0) or nine (9).
- Does the server use DYNAMIC or STATIC IP numbers?
  If the server uses STATIC IP numbers, then you need to know what IP number to use for your end of the PPP connection.
  Most Internet Service Providers use DYNAMIC IP numbers. As mentioned above, this has some implications in terms of the services you can use.
- If you are using static IP numbers, ask for the network mask your ISP uses as well.

- What are the IP numbers of the ISPs Domain Name Servers?
  There should be at least two although only one is needed.
- Does the server require the use of PAP/CHAP?
  If this is the case you need to know the "id" and "secret" you are to use in connecting. (These are probably your username and password).
- Does the server automatically start PPP or do you need to issue any commands to start PPP on the server once you are logged in?
  If you must issue a command to start PPP, what is it?

Carefully note down this information - you are going to use it!

7.1 Testing your Modem Connection for outgoing calls

You should make sure that your modem is correctly set up and that you know which serial port it is connected to.

Remember:-

- DOS com1: = Linux /dev/cua0 (and /dev/ttyS0)
- DOS com2: = Linux /dev/cua1 (and /dev/ttyS1)
  et cetera

Using you terminal communications package (such as minicom), dial into the PPP server you want to connect to with a PPP session.

(Note: at this stage we are **NOT** trying to make a PPP connection - just establishing that we have the right phone number and also to find out **exactly** what the server sends to us in order to get logged in and start PPP).

During this process, either capture (log to a file) the entire login process or carefully (very carefully) write down exactly what prompts the remote server gives to let you know it is time to enter your user name and password (and any other commands needed to establish the PPP connection).

It is worth dialing in at least twice - some servers change their prompts (e.g. with the time!) every time you log in. The two critical prompts your Linux box needs to be able to identify every time you dial in are:-

- the prompt that requests you to enter your user name;
- the prompt that requests you to enter your password;

If you have to issue a command to start PPP on the server, you will also need to find out the prompt the server gives you once you are logged in.

If your server automatically starts PPP, once you have logged in, you will start to see garbage on your screen - this is the PPP server sending your machine information to start up and configure the PPP connection.

This should look something like this :-

```
~y}#.!}!}!} }8}!}$}%U}"}&} } } } }%}& ...}'}"}(}"} .~~y}
```

(and it just keeps on coming!)

At this point, you can hang up your modem (usually, type +++ quickly and then issue the ATHO command once your modem responds with OK).

On some systems PPP must be explicitly started on the server. This is usually because the server has been set up to allow PPP logins and shell logins using the same username/password pair. If this is the case, issue this command once you have logged in. Again, you will see the garbage as the server end of the PPP connection starts up - so you can now hang up.

If you do NOT see the garbage on your screen when the server starts up PPP, it is quite likely (though not certain) that you have done something wrong.

Notwithstanding this, some PPP servers are set up to be passive - they send nothing until the client (your computer) starts the PPP process from your end.

However, the majority of servers are active and you should see the garbage.

If you can't get your modem to work, read your modem manual, the man pages for your communications software and the Serial HOWTO! Once you have this sorted out, carry on as above.

8 A note about serial ports and speed capabilities

If you are using a high speed (external) modem (14,400 Baud or above), your serial port needs to be capable of handling the throughput that such a modem is capable of producing, particularly when the modems are compressing the data.

This requires your serial port to use a modern UART (Universal Asynchronous Receiver Transmitter) such as a 16550(A). If you are using an old machine (or old serial card), it is quite possible that your serial port has only an 8250 UART, which will cause you considerable problems when used with a high speed modem.

Use the command

```
setserial -a /dev/ttySx
```

to get Linux to report to you the type of UART you have. If you do not have a 16550A type UART, invest in a new serial card (available for under $50).

Note: the first versions of the 16550 UART chip had an error. This was rapidly discovered and a revision of the chip was released - the 16550A UART. A relatively small number of the faulty chips did however get into circulation. It is unlikely that you will encounter one of these but you should look for a response that says 16550A, particularly on serial cards of some vintage.

9 Configuring your modem

You will need to configure your modem correctly for PPP - to do this **READ YOUR MODEM MANUAL**! Most modems come with a **factory default setting** that selects the options required for

PPP. The minimum configuration specifies:-

- Hardware flow control (RTS/CTS) (&K3 on many Hayes modems)

Other settings (in standard Hayes commands) you should investigate are:-

- E1 Command Echo ON (required for chat to operate)
- Q0 Report result codes (required for chat to operate)
- S0=0 Auto Answer OFF (unless you want your modem to answer the phone)
- &C1 Carrier Detect ON only after connect
- &S0 Data Set Ready (DSR) always ON
- (depends) Data Terminal Ready

It is also worth while investigating how the modem's serial interface between your computer and modem operates. Most modern modems allow you to run the serial interface at a FIXED speed whilst allowing the telephone line interface to change its speed to the highest speed it and the remote modem can both handle.

This is known as split speed operation. If your modem supports this, lock the modem's serial interface to its highest available speed (usually 115,200 baud but maybe 38,400 baud for 14,400 baud modems).

Use your communications software (e.g. minicom) to find out about your modem configuration and set it to what is required for PPP. Many modems report their current settings in response to AT&V, but you should consult your modem manual. If you completely mess up the settings, you can return to sanity (usually) by issuing an AT&F - return to factory settings. (For most modem modems I have encountered, the factory settings include all you need for PPP - but you should check).

Save your modem configuration in non-volatile RAM (usually the modem command AT&W will do this - but check in your modem manual).

With the correct modem configuration already in the modem, resetting the modem will activate this. Arranging things this way considerably simplifies the chat script necessary for the PPP connection.

9.1 Note on Serial Flow Control

When data is traveling on serial communication lines, it can happen that data arrives faster than a computer can handle it (the computer may be busy doing something else - remember, Linux is a multi-user, multi- tasking operating system). In order to ensure that data is not lost (data does not over run in the input buffer and hence get lost), some method of controlling the flow of data is necessary.

There are two ways of doing this on serial lines:-

- Using hardware signals (Clear To Send/Request to Send - CTS/RTS)
- Using software signals (control S and control Q).

Whilst the latter may be fine for a terminal (text) link, data on a PPP link uses all 8 bits - and it is quite probable that somewhere in the data there will be data bytes that translate as control S and control Q. So, if a modem is set up to use software flow control, things can rapidly go berserk!

For PPP (which uses 8 bits of data) hardware flow control is vital.

10 Using PPP and root privileges

Because PPP needs to set up networking devices, change the kernel routing table and so forth, it requires root privileges to do this.

If users other than root are to set up PPP connections, the pppd program should be setuid root :-

```
-r-sr-xr-x   1 root     root        95225 Jul 11 00:27 /usr/sbin/pppd
```

If /usr/sbin/pppd is not set up this way, then **as root** issue the command:-

```
chmod u+s /usr/sbin/pppd
```

What this does is make pppd run with root privileges **even** if the binary is run by an ordinary user. This allows a normal user to run pppd with the necessary privileges to set up the network interfaces and the kernel routing table.

Programs that run 'set uid root' are potential security holes and you should be extremely cautious about making programs 'suid root'. A number of programs (including pppd) have been carefully written to minimise the danger of running suid root, so you should be safe with this one (but no guarantees).

Depending on how you want your system to operate - specifically if you want ANY user on your system to be able to initiate a PPP link, you should make your ppp-on/off scripts world read/execute. (This is probably fine if your PC is used ONLY by you).

However, if you do NOT want just anyone to be able to start up a PPP connection (for example, your children have accounts on your Linux PC and you do not want them hooking into the Internet without your supervision), you will need to establish a PPP group (edit /etc/group) and :-

- Make the ppp-on/off scripts owned by user root and group PPP
- Make the ppp-on/off scripts read/executable by group PPP

```
-rwxr-x---   1 root     PPP           587 Mar 14  1995 /usr/sbin/ppp-on
-rwxr-x---   1 root     PPP           631 Mar 14  1995 /usr/sbin/ppp-off
```

- Make the other access rights for ppp-on/off nill.
- add the users who will be firing up PPP to the PPP group in /etc/group

Even if you do this, ordinary users will STILL not be able to shut down the link under software control! Running the `ppp-off` script requires root privileges. However, any user can just turn off the modem!

On my home PC, I do **NOT** make pppd suid root. In order to start the ppp link I must then become root - and know the password! This provides me with the ability to supervise my son's access to the Internet!

11 Setting up the PPP connection files

You now need to be logged in as root to create the directories and edit the files needed to set up PPP, even if you want PPP to be accessible to all users.

PPP uses a number of files to connect and set up a ppp connection. These differ in name and location between PPP 2.1.2 and 2.2.

For PPP 2.1.2 the files are:-

```
/usr/sbin/pppd          # the ppp binary
/usr/sbin/ppp-on        # the dialer/connection script
/usr/sbin/ppp-off       # the disconnection script
/etc/ppp/options        # the options pppd uses for all connections
/etc/ppp/options.ttyXX  # the option specific to a connection on this port
```

For PPP 2.2 the files are:-

```
/usr/sbin/pppd                  # the ppp binary
/etc/ppp/scripts/ppp-on         # the dialer/connection script
/etc/ppp/scripts/ppp-on-dialer  # part 1 of the dialer script
/etc/ppp/scripts/ppp-off        # the actual chat script itself
/etc/ppp/options                # the options pppd uses for all connections
/etc/ppp/options.ttyXX  # the option specific to a connection on this port
```

As you can see, in your /etc directory there should be a ppp directory:-

```
drwxrwxr-x   2 root     root         1024 Oct  9 11:01 ppp
```

If it does not exist - create it.

If the directory already existed, it should contain a template options file called **options.tpl**. This file is included below.

Print it out as it contains an explanation of all the PPP options (these are useful to read in conjunction with the pppd man pages). Whilst you can use this file as the basis of your `/etc/ppp/options` file, it is probably better to create your own options file that does not include all the comments in the template - it will be much shorter and easier to read/maintain.

If you have multiple serial lines/modems (typically the case for PPP servers), create a general `/etc/ppp/options` file containing the options that are common for all the serial ports on which you are supporting dial in and set up individual option files for each serial line on which you will be establishing a PPP connection with the individual settings required for each port.

These are named `options.ttyx1<`, `options.ttyx2` and so forth (where x is the appropriate letter for your serial ports).

However, for a single PPP connection, you can happily use the `/etc/ppp/options` file. Alternatively, you can put all the options as arguments in the pppd command itself.

It is easier to maintain a setup that uses `/etc/ppp/options.ttySx` files. If you use PPP to connect to a number of different sites, you can create option files for each site in `/etc/ppp/options.site` and then specify the option file as a parameter to the PPP command as you connect.

11.1 The supplied options.tpl file

Some distributions of PPP seem to have lost the options.tpl file, so here is the complete file. I suggest that you do NOT edit this file to create your `/etc/ppp/options` file(s). Rather, copy this to a new file and then edit that. If you mess up your edits, you can then go back to the original ands start again.

```
# /etc/ppp/options -*- sh -*- general options for pppd
# created 13-Jul-1995 jmk
# autodate: 01-Aug-1995
# autotime: 19:45

# Use the executable or shell command specified to set up the serial
# line.  This script would typically use the "chat" program to dial the
# modem and start the remote ppp session.
#connect "echo You need to install a connect command."

# Run the executable or shell command specified after pppd has
# terminated the link.  This script could, for example, issue commands
# to the modem to cause it to hang up if hardware modem control signals
# were not available.
#disconnect "chat -- \d+++\d\c OK ath0 OK"

# async character map -- 32-bit hex; each bit is a character
# that needs to be escaped for pppd to receive it.  0x00000001
# represents '\x01', and 0x80000000 represents '\x1f'.
#asyncmap 0

# Require the peer to authenticate itself before allowing network
# packets to be sent or received.
#auth

# Use hardware flow control (i.e. RTS/CTS) to control the flow of data
# on the serial port.
#crtscts

# Use software flow control (i.e. XON/XOFF) to control the flow of data
# on the serial port.
#xonxoff

# Add a default route to the system routing tables, using the peer as
```

```
# the gateway, when IPCP negotiation is successfully completed.  This
# entry is removed when the PPP connection is broken.
#defaultroute

# Specifies that certain characters should be escaped on transmission
# (regardless of whether the peer requests them to be escaped with its
# async control character map).  The characters to be escaped are
# specified as a list of hex numbers separated by commas.  Note that
# almost any character can be specified for the escape option, unlike
# the asyncmap option which only allows control characters to be
# specified.  The characters which may not be escaped are those with hex
# values 0x20 - 0x3f or 0x5e.
#escape 11,13,ff

# Don't use the modem control lines.
#local

# Specifies that pppd should use a UUCP-style lock on the serial device
# to ensure exclusive access to the device.
#lock

# Use the modem control lines.  On Ultrix, this option implies hardware
# flow control, as for the crtscts option.  (This option is not fully
# implemented.)
#modem

# Set the MRU [Maximum Receive Unit] value to <n> for negotiation.  pppd
# will ask the peer to send packets of no more than <n> bytes. The
# minimum MRU value is 128.  The default MRU value is 1500.  A value of
# 296 is recommended for slow links (40 bytes for TCP/IP header + 256
# bytes of data).
#mru 542

# Set the interface netmask to <n>, a 32 bit netmask in "decimal dot"
# notation (e.g. 255.255.255.0).
#netmask 255.255.255.0

# Disables the default behaviour when no local IP address is specified,
# which is to determine (if possible) the local IP address from the
# hostname. With this option, the peer will have to supply the local IP
# address during IPCP negotiation (unless it specified explicitly on the
# command line or in an options file).
#noipdefault
```

```
# Enables the "passive" option in the LCP.  With this option, pppd will
# attempt to initiate a connection; if no reply is received from the
# peer, pppd will then just wait passively for a valid LCP packet from
# the peer (instead of exiting, as it does without this option).
#passive

# With this option, pppd will not transmit LCP packets to initiate a
# connection until a valid LCP packet is received from the peer (as for
# the "passive" option with old versions of pppd).
#silent

# Don't request or allow negotiation of any options for LCP and IPCP
# (use default values).
#-all

# Disable Address/Control compression negotiation (use default, i.e.
# address/control field disabled).
#-ac

# Disable asyncmap negotiation (use the default asyncmap, i.e. escape
# all control characters).
#-am

# Don't fork to become a background process (otherwise pppd will do so
# if a serial device is specified).
#-detach

# Disable IP address negotiation (with this option, the remote IP
# address must be specified with an option on the command line or in an
# options file).
#-ip

# Disable magic number negotiation.  With this option, pppd cannot
# detect a looped-back line.
#-mn

# Disable MRU [Maximum Receive Unit] negotiation (use default, i.e.
# 1500).
#-mru

# Disable protocol field compression negotiation (use default, i.e.
# protocol field compression disabled).
#-pc
```

```
# Require the peer to authenticate itself using PAP.
#+pap

# Don't agree to authenticate using PAP.
#-pap

# Require the peer to authenticate itself using CHAP [Cryptographic
# Handshake Authentication Protocol] authentication.
#+chap

# Don't agree to authenticate using CHAP.
#-chap

# Disable negotiation of Van Jacobson style IP header compression (use
# default, i.e. no compression).
#-vj

# Increase debugging level (same as -d).  If this option is given, pppd
# will log the contents of all control packets sent or received in a
# readable form.  The packets are logged through syslog with facility
# daemon and level debug. This information can be directed to a file by
# setting up /etc/syslog.conf appropriately (see syslog.conf(5)).  (If
# pppd is compiled with extra debugging enabled, it will log messages
# using facility local2 instead of daemon).
#debug

# Append the domain name <d> to the local host name for authentication
# purposes.  For example, if gethostname() returns the name porsche,
# but the fully qualified domain name is porsche.Quotron.COM, you would
# use the domain option to set the domain name to Quotron.COM.
#domain <d>

# Enable debugging code in the kernel-level PPP driver.  The argument n
# is a number which is the sum of the following values: 1 to enable
# general debug messages, 2 to request that the contents of received
# packets be printed, and 4 to request that the contents of transmitted
# packets be printed.
#kdebug n

# Set the MTU [Maximum Transmit Unit] value to <n>. Unless the peer
# requests a smaller value via MRU negotiation, pppd will request that
# the kernel networking code send data packets of no more than n bytes
# through the PPP network interface.
#mtu <n>
```

```
# Set the name of the local system for authentication purposes to <n>.
#name <n>

# Set the user name to use for authenticating this machine with the peer
# using PAP to <u>.
#user <u>

# Enforce the use of the hostname as the name of the local system for
# authentication purposes (overrides the name option).
#usehostname

# Set the assumed name of the remote system for authentication purposes
# to <n>.
#remotename <n>

# Add an entry to this system's ARP [Address Resolution Protocol]
# table with the IP address of the peer and the Ethernet address of this
# system.
#proxyarp

# Use the system password database for authenticating the peer using
# PAP.
#login

# If this option is given, pppd will send an LCP echo-request frame to
# the peer every n seconds. Under Linux, the echo-request is sent when
# no packets have been received from the peer for n seconds. Normally
# the peer should respond to the echo-request by sending an echo-reply.
# This option can be used with the lcp-echo-failure option to detect
# that the peer is no longer connected.
#lcp-echo-interval <n>

# If this option is given, pppd will presume the peer to be dead if n
# LCP echo-requests are sent without receiving a valid LCP echo-reply.
# If this happens, pppd will terminate the connection.  Use of this
# option requires a non-zero value for the lcp-echo-interval parameter.
# This option can be used to enable pppd to terminate after the physical
# connection has been broken (e.g., the modem has hung up) in
# situations where no hardware modem control lines are available.
#lcp-echo-failure <n>

# Set the LCP restart interval (retransmission timeout) to <n> seconds
# (default 3).
```

```
#lcp-restart <n>

# Set the maximum number of LCP terminate-request transmissions to <n>
# (default 3).
#lcp-max-terminate <n>

# Set the maximum number of LCP configure-request transmissions to <n>
# (default 10).
#lcp-max-configure <n>

# Set the maximum number of LCP configure-NAKs returned before starting
# to send configure-Rejects instead to <n> (default 10).
#lcp-max-failure <n>

# Set the IPCP restart interval (retransmission timeout) to <n>
# seconds (default 3).
#ipcp-restart <n>

# Set the maximum number of IPCP terminate-request transmissions to <n>
# (default 3).
#ipcp-max-terminate <n>

# Set the maximum number of IPCP configure-request transmissions to <n>
# (default 10).
#ipcp-max-configure <n>

# Set the maximum number of IPCP configure-NAKs returned before starting
# to send configure-Rejects instead to <n> (default 10).
#ipcp-max-failure <n>

# Set the PAP restart interval (retransmission timeout) to <n> seconds
# (default 3).
#pap-restart <n>

# Set the maximum number of PAP authenticate-request transmissions to
# <n> (default 10).
#pap-max-authreq <n>

# Set the CHAP restart interval (retransmission timeout for
# challenges) to <n> seconds (default 3).
#chap-restart <n>

# Set the maximum number of CHAP challenge transmissions to <n>
# (default 10).
```

```
#chap-max-challenge

# If this option is given, pppd will rechallenge the peer every <n>
# seconds.
#chap-interval <n>

# With this option, pppd will accept the peer's idea of our local IP
# address, even if the local IP address was specified in an option.
#ipcp-accept-local

# With this option, pppd will accept the peer's idea of its (remote) IP
# address, even if the remote IP address was specified in an option.
#ipcp-accept-remote
```

11.2 What options should I use?

Well, as in all things that depends (sigh).

Provided here are two basic versions of the options file that cover the most common cases.

However, if it does NOT work, READ THE TEMPLATE FILE (`/etc/ppp/options.tpl`) **and** the pppd man pages **and** speak to the sysadmin/user support people who run the server into which you are connecting.

11.2.1 /etc/ppp/options (NO PAP/CHAP)

The following should work for connections that do not require PAP/CHAP authentication.

```
# /etc/ppp/options (NO PAP/CHAP)
#
# Prevent pppd from forking into the background
-detach
# If you are using a STATIC IP number, edit the 0.0.0.0 part of the
# following line to your static IP number.
0.0.0.0:
#
# use the modem control lines
modem
# use uucp style locks to ensure exclusive access to the serial device
lock
# use hardware flow control
crtscts
# create a default route for this connection in the routing table
defaultroute
```

```
# do NOT set up any "escaped" control sequences
asyncmap 0
# use a maximum transmission packet size of 552 bytes
mtu 552
# use a maximum receive packet size of 552 bytes
mru 552
#
#-------END OF SAMPLE /etc/ppp/options (no PAP/CHAP)
```

11.2.2 /etc/ppp/options (using PAP/CHAP)

If the server to which you are connecting requires PAP or CHAP authentication, to the above options file, add the following lines

```
#
# force pppd to use your ISP username as your 'host name' during the
# authentication process
name <your ISP username>        # you need to edit this line
#
# If you need to force PAP or CHAP authentication on the server,
# uncomment the appropriate one of the following lines.
#+chap
#+pap
#
# If you are using ENCRYPTED secrets in the /etc/ppp/pap-secrets
# file, then uncomment the following line.
#+papcrypt
```

12 Setting up your /etc/resolv.conf file

Whilst we humans like to give names to things, computers really like numbers. On a TCP/IP network (which is what the Internet is), we call machines by a particular name - and every machine lives in a particular dquot;domaindquot;. For example, my Linux workstation is called **archenland** and it resides in the **hedland.edu.au** domain. Its human readable address is thus archenland.hedland.edu.au. In order for this machine to be findable by other computers on the Internet, it is actually known by its IP number.

Translating (resolving) machine (and domain) names into the numbers actually used on the Internet is the business of machines that offer the Domain Name Service.

When you make a PPP connection, you need to tell your Linux machine where it can get host name to IP number (address resolution) information so that **you** can use the machine names but your **computer** can translate these to the IP numbers it needs to work.

One way is to enter every host that you want to talk to into the /etc/hosts file (which is in reality totally impossible if you are connecting to the Internet); another is to use the machine IP numbers as opposed to the names (an impossible memory task for all but the smallest LANs).

The best way is to set up Linux so that it knows where to go to get this name to number information - automatically. This service is provided by the Domain Name Server system. All that is necessary is to enter the IP numbers in your /etc/resolv.conf file.

Your PPP server sysadmin/user support people should provide you with two DNS IP numbers (only one is necessary - but two gives some redundancy in the event of failure).

Your /etc/resolv.conf should look something like :-

```
domain your.isp.domain.name
nameserver 10.25.0.1
nameserver 10.25.1.2
```

Edit this file (creating it if necessary) to represent the information that your ISP has provided. It should have ownership and permissions as follows :-

```
-rw-r--r--   1 root     root           73 Feb 19 01:46 /etc/resolv.conf
```

If you have already set up a /etc/resolv.conf because you are on a LAN, simply add the IP numbers of the PPP DNS servers to your existing file.

13 The PAP/CHAP secrets file

If you are using pap or chap authentication, then you also need to create the secrets file. These are:

```
/etc/ppp/pap-secrets
/etc/pp/chap-secrets
```

The first point to note about PAP and CHAP is that they are designed to authenticate **computer systems** not **users**.

"Huh? What's the difference?" I hear you ask.

Well now, once your computer has made its PPP connection to the server, **ANY** user on your system can use that connection - not just you. This is why you can set up a WAN (wide area network) link that joins two LANs (local area networks) using PPP.

That being said, your ISP will probably have given you a username and password to allow you to connect to their system and thence the Internet. Your ISP is not interested in your computer's name at all, so you will probably need to use the username at your ISP as the name for your computer.

This is done using the `name username` option to pppd. So, if you are to use the username given you by your ISP, add the line

```
name your_username_at_your_ISP
```

to your `/etc/ppp/options` file.

Technically, you should really use `user our_username_at_your_ISP` for PAP, but pppd is sufficiently intelligent to interpret `name` as `user` if it is required to use PAP. The advantage of using the `name` option is that this is also valid for CHAP.

As PAP/CHAP are for authenticating **computers**, technically you need also to specify a remote computer name. However, as most people only have one ISP, you can use a wild card (*) for the remote host name in the secrets file.

It is also worth noting that many ISPs operate multiple modem banks connected to different terminal servers - each with a different name, but ACCESSED from a single (rotary) dial in number. It can therefore be quite difficult in some circumstances to know ahead of time what the name of the remote computer is!

13.1 The PAP secrets file

The `/etc/ppp/pap-secrets` file looks like

```
# Secrets for authentication using PAP
# client        server       secret     acceptable local IP addresses
```

The four fields are white space delimited.

Suppose your ISP gave you a username of `fred` and a password of `flintstone` you would set the `name fred` option in `/etc/ppp/options.ttySx` and set up your `/etc/ppp/pap-secrets` file as follows

```
# Secrets for authentication using PAP
# client        server  secret          acceptable local IP addresses
fred            *       flintstone
```

This says for the local machine name `fred` (which we have told pppd to use even though it is not our local machine name) and for **ANY** server, use the password (secret) of `flintstone`.

Note that we do not need to specify a local IP address, unless we are required to FORCE a particular local, static IP address.

If you have several machines to which you connect using PAP, either arrange to have different usernames on each machine or find out the remote machine name to which you will be connecting. This will allow you to add lines to your `pap-secrets` file - provided you correctly set the `name` option for each separate machine to which you connect.

13.2 The CHAP secrets file

The current pppd version requires that you have mutual authentication methods - that is you must allow for both your machine to authenticate the remote server **AND** the remote server to authenticate your machine.

So, if your machine is `fred` and the remote is `barney`, your machine would set `name fred remotename barney` and the remote machine would set `name barney remotename fred` in their respective `/etc/ppp/options.ttySx` files.

The `/etc/chap-secrets` file for fred would look like

```
# Secrets for authentication using CHAP
# client        server  secret            acceptable local IP addresses
fred            barney  flintstone
```

and for barney

```
# Secrets for authentication using CHAP
# client        server  secret            acceptable local IP addresses
barney          fred    flintstone
```

14 Setting up the PPP connection manually

Now that you have created your `/etc/ppp/options` and `/etc/resolv.conf` files (and, if necessary, the `/etc/ppp/pap|chap-secrets` file), you can test the settings by manually establishing a PPP connection. (Once we have the manual connection working, we will automate the process).

To do this, your communications software must be capable of quitting WITHOUT resetting the modem. Minicom can do this - ALT Q (or in older version of minicom CTRL A Q)

Make sure you are logged in as root.

Fire up you communications software (such as minicom), dial into the PPP server and log in as normal. If you need to issue a command to start up PPP on the server, do so. You will now see the garbage you saw before.

If you are using pap/chap, then merely connecting to the remote system should start ppp on the remote and you will see the garbage without logging in (although this may not happen for some servers).

Now quit the communications software without resetting the modem (ALT Q or CTL A Q in minicom) and at the Linux prompt (as root) type

```
pppd -d -detach /dev/cuaX &
```

The -d option turns on debugging - the ppp connection start up "conversation" will be logged to your system log - which is useful if you are having trouble.

Naturally, you should use `cua0` or `cua1` etc - the actual port to which your modem is connected, **NOT** cuaX!

Your modem lights should now flash as the PPP connection is established. It will take a short while for the PPP connection to be made.

At this point you can look at the PPP interface, by issuing the command

```
ifconfig
```

In addition to any Ethernet and loop back devices you have, you should see something like :-

```
ppp0      Link encap:Point-Point Protocol
          inet addr:10.144.153.104  P-t-P:10.144.153.51 Mask:255.255.255.0
          UP POINTOPOINT RUNNING  MTU:552  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
```

Where

- inet addr:10.144.153.10 is the IP number of your end of the link.
- P-t-P:10.144.153.5 is the SERVER's IP number.

(Naturally, ifconfig will not report these IP numbers, but the ones used by your PPP server.)

Note: ifconfig also tells you that the link is UP and RUNNING!

If you get something like

```
ppp0      Link encap:Point-Point Protocol
          inet addr:0.0.0.0  P-t-P:0.0.0.0  Mask:0.0.0.0
          POINTOPOINT  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
```

Your PPP connection has not been made...see the later section on debugging!

You should also be able to see a route to the the remote host (and beyond). To do this, issue the command

```
route -n>
```

You should se something like:-

```
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
10.144.153.3    *               255.255.255.255 UH    1500   0        1 ppp0
127.0.0.0       *               255.0.0.0       U     3584   0       11 lo
10.0.0.0        *               255.0.0.0       U     1500   0       35 eth0
default         10.144.153.3    *               UG    1500   0        5 ppp0
```

Of particular importance here, notice we have TWO entries pointing to our ppp interface.

The first is a HOST route (indicated by the H flag) and that allows us to see the host to which we are connected to - but no further.

The second is the `default` route - this is the route that tells our Linux PC to send any packets NOT destined for the local Ethernet(s) - to which we have specific network routes - to the PPP server itself. The PPP server then is responsible for routing our packets out onto the Internet and routing the return packets back to us.

If you do not see a routing table with two entries, something is wrong (see the debugging section).

Now test the link by 'pinging' the server at its IP number as reported by the ifconfig output, i.e.

```
ping 10.144.153.51
```

You should receive output like

```
PING 10.144.153.51 (10.144.153.51): 56 data bytes
64 bytes from 10.144.153.51: icmp_seq=0 ttl=255 time=328.3 ms
64 bytes from 10.144.153.51: icmp_seq=1 ttl=255 time=190.5 ms
64 bytes from 10.144.153.51: icmp_seq=2 ttl=255 time=187.5 ms
64 bytes from 10.144.153.51: icmp_seq=3 ttl=255 time=170.7 ms
```

This listing will go on for ever - to stop it press CTRL C, at which point you will receive some more information :-

```
--- 10.144.153.51 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 170.7/219.2/328.3 ms
```

So far so good.

Now try pinging a host by name (not the name of the PPP server itself) but a host at another site that you KNOW is probably going to be up and running...). For example

```
ping sunsite.unc.edu
```

This time there will be a bit of a pause as Linux obtains the IP number for the fully qualified host name you have 'ping'ed from the DNS you specified in `/etc/resolv.conf` - so don't worry (but you will see your modem lights flash). Shortly you will receive output like

```
 PING sunsite.unc.edu (152.2.254.81): 56 data bytes
64 bytes from 152.2.254.81: icmp_seq=0 ttl=254 time=190.1 ms
64 bytes from 152.2.254.81: icmp_seq=1 ttl=254 time=180.6 ms
64 bytes from 152.2.254.81: icmp_seq=2 ttl=254 time=169.8 ms
64 bytes from 152.2.254.81: icmp_seq=3 ttl=254 time=170.6 ms
64 bytes from 152.2.254.81: icmp_seq=4 ttl=254 time=170.6 ms
```

Again, stop the output by pressing CTRL C and get the statistics...

```
--- sunsite.unc.edu ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 169.8/176.3/190.1 ms
```

If you don't get any response, check in the debugging section of this document.

If everything works, shut down the connection by typing

```
ppp-off
```

After a short pause, the modem should hang itself up.

If that does not work, either turn off your modem or fire up your communications software and interrupt the modem with +++ and then hang up with ATH0 when you receive the modem's OK prompt.

You may also need to clean up the lock file created by pppd

```
rm -f /var/lock/LCK..cuaX
```

15 Automating your connections - Creating the connection scripts

A chat script automates the log in and PPP start up so all you have to do (as root or as a member of the ppp group) is issue a single command to fire up your connection.

15.1 Connection scripts for Username/Password Authentication

If your ISP does NOT require the use of PAP/CHAP, these are the scripts for you!

If the ppp package installed correctly, you should have two example files. For PPP 2.1.2 they are in `/usr/sbin` and for PPP 2.2 they are in `/etc/ppp/scripts`. They are called

for PPP-2.1.2

```
ppp-on
ppp-off
```

and for PPP-2.2

```
ppp-off
ppp-on
ppp-on-dialer
```

Now, if you are using PPP 2.1.2, I strongly urge you to delete the sample files. There are potential problems with these - and don't tell me they work fine - I used them for ages too (and recommended them in the first version of this HOWTO!

For the benefit of PPP 2.1.2 users, here are BETTER template versions, taken from the PPP 2.2 distribution. I suggest you copy and use these scripts **instead of** the old PPP-2.1.2 scripts.

15.2 The ppp-on script

This is the first of a PAIR of scripts that actually fire up the connection.

```
#!/bin/sh
#
# Script to initiate a PPP connection. This is the first part of the
# pair of scripts. This is not a secure pair of scripts as the codes
# are visible with the 'ps' command.  However, it is simple.
#
# These are the parameters. Change as needed.
TELEPHONE=555-1212      # The telephone number for the connection
ACCOUNT=george          # The account name for logon (as in 'George Burns')
PASSWORD=gracie         # The password for this account (and 'Gracie Allen')
LOCAL_IP=0.0.0.0        # Local IP address if known. Dynamic = 0.0.0.0
REMOTE_IP=0.0.0.0       # Remote IP address if desired. Normally 0.0.0.0
NETMASK=255.255.255.0   # The proper netmask if needed
#
# Export them so that they will be available to 'ppp-on-dialer'
export TELEPHONE ACCOUNT PASSWORD
#
# This is the location of the script which dials the phone and logs
# in.  Please use the absolute file name as the $PATH variable is not
# used on the connect option.  (To do so on a 'root' account would be
# a security hole so don't ask.)
#
DIALER_SCRIPT=/etc/ppp/ppp-on-dialer
#
# Initiate the connection
#
#
exec /usr/sbin/pppd debug /dev/ttySx 38400 \
        $LOCAL_IP:$REMOTE_IP \
        connect $DIALER_SCRIPT
```

Here is the ppp-on-dialer script:-

```
#!/bin/sh
#
# This is part 2 of the ppp-on script. It will perform the connection
# protocol for the desired connection.
#
chat -v                                                 \
        TIMEOUT         3                               \
```

```
ABORT           '\nBUSY\r'                      \
ABORT           '\nNO ANSWER\r'                \
ABORT           '\nRINGING\r\n\r\nRINGING\r'   \
''              \rAT                           \
'OK-+++\c-OK'   ATH0                           \
TIMEOUT         30                             \
OK              ATDT$TELEPHONE                 \
CONNECT         ''                             \
ogin:--ogin:    $ACCOUNT                       \
assword:        $PASSWORD
```

15.3 Editing the supplied PPP startup scripts

As the new scripts come in two parts, we will edit them in turn.

15.3.1 The ppp-on script

You will need to edit the script to reflect YOUR user name at your ISP, YOUR password at your ISP, the telephone number of your ISP.

Each of the lines like `TELEPHONE=` actually set up shell variables that contain the information to the right of the '=' (excluding the comments of course). So edit each of these lines so it is correct for your ISP and connection.

Also, as you are setting the IP number (if you need to) in the `/etc/ppp/options` file, DELETE the line that says

```
$LOCAL_IP:$REMOTE_IP \
```

Also, make sure that the shell variable DIALER_SCRIPT points at the full path and name of the dialer script that you are actually going to use. So, if you have moved this or renamed the script, make sure you edit this line correctly in the `ppp-on` script!

If you have set up your `ppp-on` script correctly and your PPP server uses username/password authentication, you should not need to edit the `ppp-on-dialer` script at all!

Although you can set up your serial port using /etc/rc.serial at boot time, I have found that it is a good idea to explicitly set up the serial port in the ppp-on script. This allows for my using the modem for other purposes (which may reset the serial settings) between times.

Immediately before the line that actually starts pppd, add the line

```
/bin/setserial /dev/cuaX spd_vhi
```

This sets up the serial port to actually set the baud rate to 115,200 baud when a speed of 38,400 baud is requested. This is fine for 28.8k (and faster) baud modems. However, many 14,400 baud modems cannot actually run their serial interface back to the computer at this speed.

Check you modem manual and if the maximum serial speed for your modem is 38,400 use the line

```
/bin/setserial /dev/cuaX spd_normal
```

15.3.2 Starting PPP at the server end

Whilst the ppp-on-dialer script is fine for servers that automatically start pppd at the server end once you have logged in, some servers require that you explicitly start PPP on the server.

If you need to issue a command to start up PPP on the server, you DO need to edit the ppp-on-dialer script.

At the END of the script (after the password line) add an additional **expect send** pair - this one would look for your login prompt (beware of characters that have a special meaning in the Bourne shell - such as $ and *or* (open and close square brackets).

Once chat has found the shell prompt, chat must issue the ppp start up command required for your ISPs PPP server.

In my case, my PPP server uses the standard Linux Bash prompt

```
[hartr@kepler hartr]$
```

and requires that I type

```
ppp
```

to start up PPP on the server.

It is a good idea to allow for a bit of error recovery here, so in my case I use

```
        hartr--hartr    ppp
```

This says - if we don't receive the prompt within the timeout, send a carriage return and looks for the prompt again.

Once the prompt is received, then send the string 'ppp'.

Note: don't forget to add a to the end of the previous line so chat still thinks the entire chat script is on one line!

Unfortunately, some servers produce a very variable set of prompts! You may need to log in several times using minicom to understand what is going on and pick the stable "expect" strings.

15.3.3 The ppp-on-dialer script

This is the second of the scripts that actually brings up our ppp link.

Note: a chat script is normally all on one line. the backslashes are used to allow line continuations across several physical lines (for human readability) and do not form part of the script itself.

However, it is very useful to look at it in detail so that we understand what it is actually (supposed) to be doing!

15.4 What a Chat script means...

A chat script is a sequence of "expect string" "send string" pairs. In particular, note that we **ALWAYS** expect **something** before we send something.

If we are to send something **WITHOUT** receiving anything first, we must use an empty expect string (indicated by "") and similarly for expecting something without sending anything! Also, if a string consists of several words, (e.g. NO CARRIER), you must quote the string so that it is seen as a single entity by chat.

The chat line in our template is:-

- `exec chat -v`
  Invoke chat, the -v tells chat to copy ALL its I/O into the system log (usually /var/log/messages). Once you are happy that the chat script is working reliably, edit this line to remove the -v to save unnecessary clutter in your syslog.
- `TIMEOUT 3`
  This sets the timeout for the receipt of expected input to three seconds. You may need to increase this to say 5 or 10 seconds if you are using a really slow modem!
- `ABORT '\nBUSY\r'`
  If the string BUSY is received, abort the operation.
- `ABORT '\nNO ANSWER\r'`
  If the string NO ANSWER is received, abort the operation
- `ABORT '\nRINGING\r\n\r\nRINGING\r'`
  If the (repeated) string RINGING is received, abort the operation.
  This is because someone is ringing your phone line!
- `'' \rAT`
  Expect nothing from the modem and send the string AT.
- `'OK-+++\c-OK' ATH0`
  This one is a bit more complicated as it uses some of chat's error recovery capabilities. What is says is...
  Expect OK, if it is NOT received (because the modem is not in command mode) then send +++ (the standard Hayes-compatible modem string that returns the modem to command mode) and expect OK; then send ATH0 (the modem hang up string).

This allows your script to cope with the situation of your modem being stuck on-line!

- `TIMEOUT 30`
  Set the timeout to 30 seconds for the remainder of the script. If you experience trouble with the chat script aborting due to timeouts, increase this to 45 seconds or more.
- `OK ATDT$TELEPHONE`
  Expect OK (the modem's response to the ATH0 command) and dial the number we want to call.
- `CONNECT ''`
  Expect CONNECT (which our modem sends when the remote modem answers) and send nothing in reply.
- `ogin:--ogin: $ACCOUNT`
  Again, we have some error recovery built in here. Expect the login prompt (...ogin:) but if we don't receive it by the timeout, send a return and then look for the login prompt again. When the prompt is received, send the username (stored in the shell variable $ACCOUNT).
- `assword: $PASSWORD`
  Expect the password prompt and send our password (again, stored in a shell variable).

This chat script has reasonable error recovery capability. Chat has considerably more features than demonstrated here. For more information consult the chat manual page (`man 8 chat`).

15.5 A chat script for PAP/CHAP authenticated connections

If your ISP is using PAP/CHAP, then your chat script is much simpler. All your chat script needs to do is dial the telephone, wait for a connect and then let pppd handle the logging in!

```
#!/bin/sh
#
# This is part 2 of the ppp-on script. It will perform the connection
# protocol for the desired connection.
#
exec chat -v                                            \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDT$TELEPHONE                  \
        CONNECT         ''                              \
```

15.6 The pppd debug and -f option_ file options

As we have already seen, you can turn on debug information logging with the -d option to pppd. The 'debug' option is equivalent to this.

As we are establishing a new connection with a new script, leave in the debug option for now. (Warning: if your disk space is tight, logging pppd exchanges can rapidly extend your syslog file and run you into trouble - but to do this you must fail to connect and keep on trying for quite a few minutes).

Once you are happy that all is working properly, then you can remove this option.

If you have called your ppp options file anything other than `/etc/ppp/options` or `/etc/ppp/options.ttySx`, specify the file name with the -f option to pppd - e.g.

```
exec /usr/sbin/pppd debug -f options.myserver /dev/ttySx 38400 \
```

16 Testing your connection script

Open a new root Xterm (if you are in X) or open a new virtual console and log in as root.

In this new session, issue the command

```
tail -f /var/log/messages
```

(or whatever your system log file is).

In the first window (or virtual console) issue the command

```
ppp-on \&
```

(or whatever name you have called your edited version of /usr/sbin/ppp- on). If you do not put the script into the background by specifying & at the end of the command, you will not get your terminal prompt back until ppp exits (when the link terminates).

Now switch back to the window that is tracking your system log.

You will see something like the following (provided you specified -v to chat and -d to pppd)....this is the chat script and responses being logged to the system log file followed by the start up information for pppd :-

```
Oct 21 16:09:58 hwin chat[19868]: abort on (NO CARRIER)
Oct 21 16:09:59 hwin chat[19868]: abort on (BUSY)
Oct 21 16:09:59 hwin chat[19868]: send (ATZ^M)
Oct 21 16:09:59 hwin chat[19868]: expect (OK)
Oct 21 16:10:00 hwin chat[19868]: ATZ^M^M
Oct 21 16:10:00 hwin chat[19868]: OK -- got it
Oct 21 16:10:00 hwin chat[19868]: send (ATDT722298^M)
Oct 21 16:10:00 hwin chat[19868]: expect (CONNECT)
```

```
Oct 21 16:10:00 hwin chat[19868]: ^M
Oct 21 16:10:22 hwin chat[19868]: ATDT722298^M^M
Oct 21 16:10:22 hwin chat[19868]: CONNECT -- got it
Oct 21 16:10:22 hwin chat[19868]: send (^M)
Oct 21 16:10:22 hwin chat[19868]: expect (ogin:)
Oct 21 16:10:22 hwin chat[19868]:  57600^M
Oct 21 16:10:23 hwin chat[19868]: ^[[;H^[[2J^M^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I      This is node  kepler.hedland.edu.au^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I^I      at Hedland Campus^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I^I       Hedland College^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I  Authorised user ONLY are to use this system^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I For more information, contact ComputerSystems^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I^I     on +61 (0)91 72 0400^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I^I^I     or^M
Oct 21 16:10:23 hwin chat[19868]: ^I^I       email:  help@hedunx.hedland.edu.au^M
Oct 21 16:10:23 hwin chat[19868]: ^M
Oct 21 16:10:23 hwin last message repeated 3 times
Oct 21 16:10:23 hwin chat[19868]: kepler login: -- got it
Oct 21 16:10:23 hwin chat[19868]: send (hartr^M)
Oct 21 16:10:23 hwin chat[19868]: expect (ssword:)
Oct 21 16:10:23 hwin chat[19868]:  hartr^M
Oct 21 16:10:23 hwin chat[19868]: Password: -- got it
Oct 21 16:10:23 hwin chat[19868]: send (??????^M)
Oct 21 16:10:23 hwin chat[19868]: expect (hartr)
Oct 21 16:10:23 hwin chat[19868]:  ^M^M
Oct 21 16:10:24 hwin chat[19868]: Last login: Sat Oct 21 14:55:53 on  ttyC0^M
Oct 21 16:10:24 hwin chat[19868]: ^M
Oct 21 16:10:24 hwin last message repeated 9 times
Oct 21 16:10:24 hwin chat[19868]: ^I^IYou have logged into node  kepler.hedland.edu.au^M
Oct 21 16:10:24 hwin chat[19868]: ^M
Oct 21 16:10:24 hwin chat[19868]:            This is a Compaq Prolinea 486DX2/50 running Linux 1.1.54^M
Oct 21 16:10:24 hwin chat[19868]: ^M
Oct 21 16:10:24 hwin chat[19868]:        This computer operates as the main Hedland Campus communicatio
Oct 21 16:10:24 hwin chat[19868]: ^I       node, providing dial-in terminal and SLIP access,^M
Oct 21 16:10:24 hwin chat[19868]: ^I^I      Kepler also runs the Hedland end of^M
Oct 21 16:10:24 hwin chat[19868]: ^I^I    the Hedland/Newman inter-Campus WAN link^M
Oct 21 16:10:24 hwin chat[19868]: ^M
Oct 21 16:10:24 hwin chat[19868]: ^M
Oct 21 16:10:24 hwin chat[19868]: [hartr -- got it
```

```
Oct 21 16:10:24 hwin chat[19868]: send (ppp^M)
Oct 21 16:10:27 hwin pppd[19872]: pppd 2.1.2 started by root, uid 0
Oct 21 16:10:27 hwin pppd[19873]: Using interface ppp0
Oct 21 16:10:27 hwin pppd[19873]: Connect: ppp0 <--> /dev/cua1
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(LCP): Sent code 1, id 1.
Oct 21 16:10:27 hwin pppd[19873]: LCP: sending Configure-Request, id 1
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfreq(LCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd MRU
Oct 21 16:10:27 hwin pppd[19873]: (1500)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd ASYNCMAP
Oct 21 16:10:27 hwin pppd[19873]: (0)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd MAGICNUMBER
Oct 21 16:10:27 hwin pppd[19873]: (a098b898)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd PCOMPRESSION
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd ACCOMPRESSION
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: returning CONFACK.
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(LCP): Sent code 2, id 1.
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfack(LCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 1, id 1.
Oct 21 16:10:27 hwin pppd[19873]: IPCP: sending Configure-Request, id 1
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfreq(IPCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: ipcp: received ADDR
Oct 21 16:10:27 hwin pppd[19873]: (10.144.153.51)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: ipcp: received COMPRESSTYPE
Oct 21 16:10:27 hwin pppd[19873]: (45)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: ipcp: returning Configure-ACK
Oct 21 16:10:28 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 2, id 1.
Oct 21 16:10:30 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 1, id 1.
Oct 21 16:10:30 hwin pppd[19873]: IPCP: sending Configure-Request, id 1
Oct 21 16:10:30 hwin pppd[19873]: fsm_rconfreq(IPCP): Rcvd id 255.
Oct 21 16:10:31 hwin pppd[19873]: ipcp: received ADDR
Oct 21 16:10:31 hwin pppd[19873]: (10.144.153.51)
Oct 21 16:10:31 hwin pppd[19873]:  (ACK)
Oct 21 16:10:31 hwin pppd[19873]: ipcp: received COMPRESSTYPE
Oct 21 16:10:31 hwin pppd[19873]: (45)
Oct 21 16:10:31 hwin pppd[19873]:  (ACK)
Oct 21 16:10:31 hwin pppd[19873]: ipcp: returning Configure-ACK
```

```
Oct 21 16:10:31 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 2, id 255.
Oct 21 16:10:31 hwin pppd[19873]: fsm_rconfack(IPCP): Rcvd id 1.
Oct 21 16:10:31 hwin pppd[19873]: ipcp: up
Oct 21 16:10:31 hwin pppd[19873]: local  IP address 10.144.153.104
Oct 21 16:10:31 hwin pppd[19873]: remote IP address 10.144.153.51
```

(Note - I am using STATIC IP numbers - hence my machine sent that to the PPP server - you won't see this if you are using DYNAMIC IP numbers.)

This looks OK - so test it out as before with pings to IP numbers and host names.

Fire up you web browser or whatever and go surfing - you are connected!

17 Shutting down the PPP link

When you have finished with the PPP link, use the standard ppp-off command to shut it down (remember - you need to be root or a member of the PPP group!).

In your system log you will see something like:-

```
Oct 21 16:10:45 hwin pppd[19873]: Interrupt received: terminating link
Oct 21 16:10:45 hwin pppd[19873]: ipcp: down
Oct 21 16:10:45 hwin pppd[19873]: default route ioctl(SIOCDELRT): Bad address
Oct 21 16:10:45 hwin pppd[19873]: fsm_sdata(LCP): Sent code 5, id 2.
Oct 21 16:10:46 hwin pppd[19873]: fsm_rtermack(LCP).
Oct 21 16:10:46 hwin pppd[19873]: Connection terminated.
Oct 21 16:10:46 hwin pppd[19873]: Exit.
```

Don't worry about the `SIOCDELRT` - this is just pppd noting that it is terminating and is nothing to worry about.

18 Debugging

There are any number of reasons that your connection does not work - chat has failed to complete correctly, you have a dirty line, etc. So check your syslog for indications.

A **VERY** common mistake is that you have mistyped something in your scripts. You need to check these through very carefully - and bear in mind that we humans have a tendency to read what we THINK we have typed - not what is actually there!

Another is to try to use PPP-2.2 with kernel 1.2.X or PPP-2.1.2 with kernel 1.3.X/2.0.X - use the right version of pppd for your kernel!

Now look in the PPP FAQ (which is really a series of questions and answers). This is a very comprehensive document and the answers ARE there! From my own (sad) experience, if the answer to your problems is

not there, the problem is NOT ppp's fault! In my case I was using an ELF kernel that I had not upgraded to the appropriate kernel modules. I only wasted about 2 days (and most of one night) cursing what had been a perfect PPP server before the light dawned!

18.1 I compiled in PPP but Linux says I don't have it!

You are using kernel 1.3.X/2.0.X and have compiled in module support and then compiled PPP support as a module (and installed the modules) - haven't you?

If you are NOT using kerneld to autoload the required modules, then you must explicitly load the ppp module (and possibly the serial support module too) before you can run PPP!

You can do this by hand - as root, type

```
insmod ppp
```

You may also need to load the serial support module first...

```
insmod slhc
```

However, you should sort out the auto-loading of kernel modules - so go check out the kerneld mini-howto!

Another possibility is that you are trying to use ppp-2.1.2 with Linux kernel 2.0.x (or alternatively that you are using ppp-2.2 with Linux kernel 1.2.x and haven't made the patches necessary to the kernel).

Check your versions of kernel and ppp! To re-iterate:-

Linux kernel version 2.0.x REQUIRES ppp-2.2.

bf/Linux kernel version 1.2.x works with ppp-2.1.2 but can be made to work with ppp-2.2./

18.2 I cannot set up a default route

You have a local Ethernet (or another network connection of some kind) with an existing default route already set up.

The section on routing in 'Linking two networks using PPP' covers correctly setting this up (briefly).

Your problem is that you cannot have more than a single default route. A default route is the destination to which all packets are sent that are not covered by a **specific** route. Generally, the default route will point at the route from your computer to the Internet.

Unfortunately, some Linux distributions set up a default route to the local Ethernet interface. You will need to change the system initialisation that handles configuring your Ethernet interface and establishes the routing across that interface so that it sets up a specific route to your local Ethernet(s).

See the NET2-Howto and the Linux Network Administrator Guide for this information.

19 Linking two networks using PPP

There is basically no difference between linking a single Linux PC to a PPP server and linking two LANs using PPP on a machine on each LAN. Remember, PPP is a **peer to peer** protocol.

However, you **DEFINITELY** need to understand about how routing is established. Read the NET-2 howto and the Linux Network Administrator Guide (NAG). You will also find " TCP/IP Network Administration" (published by O'Reilly and Assoc - ISBN 0-937175-82-X) to be of invaluable assistance.

In order to link two LANs, you **must** be using different IP network numbers (or subnets of the same network number) and you will need to use static IP numbers - or use IP masquerade. If you want to use IP masquerade, see the IP masquerade mini-howto for instructions on setting that up.

19.1 Setting up IP numbers

Arrange with the network administrator of the other LAN the IP numbers that will be used for each end of the PPP interface. If you are using static IP numbers, this will also probably require you to dial into a specific telephone number.

Now edit the appropriate `/etc/ppp/options[.ttyXX]` file - it's a good idea to have a specific modem and port at your end for this connection. This may well require you to change your `/etc/ppp/options` file too - and create appropriate options.ttyXX files for any other connections too!

Specify the IP numbers for your end of the PPP link in the appropriate options file exactly as shown above for static IP numbers.

19.2 Setting up the routing

You must arrange that packets on your local LAN are routed across the interface that the PPP link establishes. This is a two stage process.

First of all, you need to establish a route from the machine running the PPP link to the network(s) at the far end of the link. If the link is to the Internet, this can be handled by a default route established by pppd itself at your end of the connection using the 'defaultroute' option to pppd.

If however, the link is only linking two LANs, then a specific network route must be added. This is done using a 'route' command in the /etc/ppp/ip-up script (see After the link comes up...) for instructions on doing this.

The second thing you need to do is to tell the other computers on your LAN that your Linux computer is actually the 'gateway' for the network(s) at the far end of the ppp link.

Of course, the network administrator at the other end of the link has to do all this too! However, as s/he will be routing packets to your specific networks, a **specific network route** will be required, not a default route (unless the LANs at the far and of the link are linking into you to access the Internet across your connection).

19.3 Network security

If you are linking you LAN to the Internet using PPP - or even just to a "foreign" LAN, you need to think about security issues. I strongly urge you to think about setting up a firewall!

20 After the link comes up...

Once the PPP link is established, pppd looks for `/etc/ppp/ip-up`. If this script exists and is executable, the PPP daemon executes the script. This allows you to automate any special routing commands that may be necessary and any other actions that you want to occur every time the PPP link is activated.

This is just a shell script and can do anything that a shell script can do (i.e. virtually anything you want).

For example, you can get sendmail to dispatch any waiting outbound messages in the mail queue.

Similarly, you can insert the commands into ip-up to collect (using pop) any email waiting for you at your ISP.

20.1 Special routing

If you are linking two LANs, you will need to set up a specific route to the 'foreign' LANs. This is easily done using the `/etc/ppp/ip-up` script. The only difficulty arises if your machine handles multiple PPP links.

This is because the `/etc/ppp/ip-up` is executed for EVERY ppp connection that comes up, so you need to carefully execute the correct routing commands for the particular link that comes up.

Using the bash 'case' statement on an appropriate parameter that pppd passes into the script accomplishes this. For example, this is the `/etc/ppp/ip-up` script I use to handle our WAN links and the link to my home Ethernet (also handled on the same ppp server).

```
#!/bin/bash
#
# Script which handles the routing issues as necessary for pppd
# Only the link to Newman requires this handling.
#
# When the ppp link comes up, this script is called with the following
# parameters
#       $1      the interface name used by pppd (e.g. ppp3)
#       $2      the tty device name
#       $3      the tty device speed
#       $4      the local IP address for the interface
#       $5      the remote IP address
#       $6      the parameter specified by the 'ipparam' option to pppd
#
case "$5" in
```

```
# Handle the routing to the Newman Campus server
        202.12.126.1)
                /sbin/route add -net 202.12.126.0 gw 202.12.126.1
# and flush the mail queue to get their email there asap!
                /usr/sbin/sendmail -q &
                ;;
        139.130.177.2)
# Our Internet link
# When the link comes up, start the time server and synchronise to the world
# provided it is not already running
                if [ ! -f /var/lock/subsys/xntpd ]; then
                        /etc/rc.d/init.d/xntpd.init start &
                fi
# Start the news server (if not already running)
                if [ ! -f /var/lock/subsys/news ]; then
                        /etc/rc.d/init.d/news start &
                fi
                ;;
        203.18.8.104)
# Get the email down to my home machine as soon as the link comes up
# No routing is required as my home Ethernet is handled by IP
# masquerade and proxyarp routing.
                /usr/sbin/sendmail -q &
                ;;
        *)
esac
exit 0
```

As a result of bringing up the ppp link to our Newman campus and this script, we end up with the following routing table entries (this machine also is our general dial up PPP server AND handles our Internet link). I have interspersed comments in the output to help explain what each entry is) :-

---

```
[root@kepler /root]# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
# the HOST route to our remote internet gateway
139.130.177.2   *               255.255.255.255 UH    1500   0      134 ppp4
# the HOST route to our Newman campus server
202.12.126.1    *               255.255.255.255 UH    1500   0       82 ppp5
# the HOST route to my home ethernet
203.18.8.104    *               255.255.255.255 UH    1500   0       74 ppp3
# two of our general dial up PPP lines
203.18.8.64     *               255.255.255.255 UH    552    0        0 ppp2
203.18.8.62     *               255.255.255.255 UH    552    0        1 ppp1
```

```
# the specific network route to the Newman campus LAN
202.12.126.0    202.12.126.1    255.255.255.0   UG    1500   0        0 ppp5
# the route to our local ethernet (super-netting two adjacent C classes)
203.18.8.0      *               255.255.254.0   U     1500   0     1683 eth0
# the route to the loop back device
127.0.0.0       *               255.0.0.0       U     3584   0      483 lo
# the default route to the Internet
default         139.130.177.2   *               UG    1500   0     3633 ppp4
```

20.2 Handling email

The previous section shows how to handle the outgoing mail - simply by flushing the mail queue once the link is up.

If you are running a WAN link, you can arrange with the network administrator of the remote LAN to do exactly the same thing. For example, at the Newman Campus end of our WAN link, the `/etc/ppp/ip-up` script looks like :-

```
#!/bin/bash
#
# Script which handles the routing issues as necessary for pppd
# Only the link to Hedland requires this handling.
#
# When the ppp link comes up, this script is called with the following
# parameters
#       $1      the interface name used by pppd (e.g. ppp3)
#       $2      the tty device name
#       $3      the tty device speed
#       $4      the local IP address for the interface
#       $5      the remote IP address
#       $6      the parameter specified by the 'ipparam' option to pppd
#
case "$5" in
        203.18.8.4)
                /usr/sbin/sendmail -q
                ;;
        *)
esac
exit 0
```

If however you have only a dynamic IP PPP link to your ISP, you need to get your email from the account on your ISPs machine. This is usually done using the POP (Post Office Protocol). This process can be handled using the 'popclient' program - and the ip-up script can automate this process for you too!

Simply create a `/etc/ppp/ip-up` script that contains the appropriate invocation of popclient. For my laptop that runs Red Hat Linux (which I take on any travels), this is

```
popclient -3 -c -u hartr -p <password> kepler.hedland.edu.au |formail -s procmail
```

You could use slurp or whatever to do the same for news, and so forth. Remember, the ip-up script is just a standard bash script and so can be used to automate ANY function that needs to be accomplished every time the appropriate PPP link comes up.

21 Shutting down the link

The existing /usr/sbin/ppp-off script should work just fine when run as user root. The only changes you may wish to make are for the script to wait for any outgoing email currently being processed by sendmail.

This is left as an exercise for the student!

In addition, you can create a script file that will be executed once the link has been terminated. This is stored in /etc/ppp/ip-down. It can be used to undo anything special that you did in the corresponding /etc/ppp/ip-up script.

22 Routing issues on a LAN

If you are connected to a LAN but still want to use PPP on your personal Linux machine , you need to address some issues of the routes packets need to take from your machine to reach your LAN (through your Ethernet interface) and also to the remote PPP server and beyond.

This section does NOT attempt to teach you about routing - it deals only with a simple, special case of (static) routing!

I strongly urge you to read the Linux Network Administrator Guide (NAG) if you are NOT familiar with routing. Also the O'Reilly book "TCP/IP Network Administration" covers this topic in a very understandable form.

The basic rule of static routing is that the DEFAULT route should be the one that points to the MOST number of network addresses. For other networks, enter specific routes to the routing table.

The ONLY situation I am going to cover here is where your Linux box is on a LAN that is not connected to the Internet - and you want to dial out to the Internet for personal use whilst still connected to the LAN.

First of all, make sure that your Ethernet route is set up to the specific network addresses available across your LAN - NOT set to the default route!

Check this by issuing a route command, you should see something like the following:-

```
[root@hwin /root]# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
```

```
loopback        *               255.255.255.0   U     1936   0       50 lo
10.0.0.0        *               255.255.255.0   U     1436   0      565 eth0
```

If your Ethernet interface (eth0) is pointing at the default route, (the first column will show "default" in the eth0 line) you need to change your Ethernet initialisation scripts to make it point at the specific network numbers rather than the default route (consult the Net2 HOWTO and NAG).

This will allow pppd to set up your default route as shown below:-

```
[root@hwin /root]# route -n
Kernel routing table

Destination     Gateway         Genmask         Flags MSS    Window Use Iface
10.144.153.51   *               255.255.255.255 UH    488    0        0 ppp0
127.0.0.0       *               255.255.255.0   U     1936   0       50 lo
10.1.0.0        *               255.255.255.0   U     1436   0      569 eth0
default         10.144.153.51   *               UG    488    0        3 ppp0
```

As you can see, we have a host route to the PPP server (10.144.153.51) via ppp0 and also a default network route that uses the PPP server as its gateway.

If your set up needs to be more complex than this - read the routing documents already mentioned and consult an expert at your site!

If your LAN already has routers on it, you will already have gateways established to the wider networks available at your site. You should STILL point your default route at the PPP interface - and make the other routes specific to the networks they serve.

22.1 Note on Security

When you set up a Linux box on an existing LAN to link into the Internet, you are potentially opening your entire LAN to the Internet - and the hackers that reside there. Before you do this, I strongly urge you to consult your network administrator and site security policy. If your PPP connection to the Internet is used to successfully attack your site, you will at the very least earn the intense anger of your fellow users, network and system administrators. You may also find yourself in very much more serious trouble!

Before you connect a LAN to the Internet, you should consider the security implications of even a DYNAMIC connection - hence the earlier reference to the O'Reilly "Building Internet Firewalls"!

23 Getting Help when totally stuck

If you can't get your PPP link to work, go back through this document and check everything - in conjunction with the output created by "chat-v..." and "pppd -d" in you system log.

Also consult the PPP documentation and FAQ plus the other documents mention herein!

If you are still stuck, try the comp.os.linux.misc and comp.os.linux.networking newsgroups are reasonably regularly scanned by people that can help you with PPP as is comp.protocols.ppp

You can try sending me personal email, but I do have a day job (and a life) and I do not guarantee to respond quickly (if at all) as this depends on my current work load and the state of my private life!

In particular - **DO NOT POST REAMS OF DEBUGGING OUTPUT TO THE NEWS GROUPS NOR SEND IT TO ME BY EMAIL** - the former wastes huge amounts of network bandwidth and the latter will be consigned to /dev/null (unless I have specifically requested it).

24 Common Problems once the link is working

One problem you will find is that many service providers will only support the connection software package that they distribute to new accounts. This is (typically) for Microsoft Windows :-(- and many service provider help desks seem to know nothing about Unix (or Linux). So, be prepared for limited assistance from them!

You could of course do the individual a favour and educate then about Linux (any ISP help desk person should be reasonably 'with it' in Internet terms and that means they should have a home Linux box - of course it does)!

24.1 I can't see beyond the PPP server I connect to

OK - your PPP connection is up and running and you can ping the PPP server by IP number (the second or "remote" IP number shown by ifcongig ppp0), but you can't reach anything beyond this.

First of all, try pinging the IP numbers you have specified in /etc/resolv.conf as name servers. If this works, you can see beyond your PPP server (unless this has the same IP number as the "remote" IP number of your connection). So now try pinging the full Internet name of your service provider - eg

```
ping my.provider.net.au
```

If this does NOT work, you have a problem with the name resolution. This is probably because of a typo in your /etc/resolv.conf file. Check this carefully against the information you acquired by ringing your service provider. If all looks OK, ring your service provider and check that you wrote down the IP numbers correctly.

If it STILL doesn't work (and your service provider confirms that his name servers are up and running), you have a problem somewhere else - and I suggest you check carefully through your Linux installation (looking particularly for file permissions).

If you STILL can't ping your service provider's IP name servers by IP number, either they are down (give them a voice call and check) or there is a routing problem at your service provider's end. Again, ring them and check this out.

One possibility is that the "remote end" is a Linux PPP server where the IP forwarding option has not been specified in the kernel!

A good general test is to try hooking in to your service provider using the software that most supply for (gulp) Microsoft Windows. If everything works from another operating system to exactly the same account, then the problem is with your Linux system and NOT your service provider.

24.2 I can send email, but not receive it

If you are using dynamic IP numbers, this is perfectly normal. See "Setting up Services" below.

24.3 Why can't people finger, WWW, gopher, talk etc to my machine?

Again, if you are using dynamic IP numbers, this is perfectly normal. See "Setting up Services" below.

25 Using Internet services with Dynamic IP numbers

If you are using dynamic IP number (and many service providers will only give you a dynamic IP number unless you pay significantly more for your connection), then you have to recognise the limitations this imposes.

First of all, outbound service requests will work just fine. That is you can send email using sendmail, ftp files from remote sites, finger users on other machines, browse the web etc.

In particular, you can answer email that you have brought down to your machine whilst you are off line. Mail will simply sit in your mail queue until you dial back into your ISP.

However, your machine is NOT connected to the Internet 24 hours a day, nor does it have the same IP number every time it is connected. So it is impossible for you to receive email directed to your machine, and very difficult to set up a web or ftp server that your friends can access! As far as the Internet is concerned your machine does not exist as a unique, permanently contactable machine as it does not have a unique IP number (remember - other machines will be using the IP number when they are allocated it on dial in).

If you set up a WWW (or any other server), it is totally unknown by any user on the Internet UNLESS they know that your machine is connected AND its actual (current) IP number. There are a number of ways they can get this info, ranging from you ringing them, sending them email to tell them or cunning use of ".plan" files on a shell account at your service provider (assuming that your provider allows shell access).

Now, for most users, this is not a problem - all that most people want is to send and receive email (using your account on your service provider) and make outbound connections to WWW, ftp and other servers on the Internet. If you MUST have inbound connections to your server, you should really get a static IP number. Alternatively you can explore the methods hinted at above...

25.1 Setting up email

Even for dynamic IP numbers, you can certainly configure sendmail on your machine to send out any email that you compose locally. Configuration of sendmail can be obscure and difficult - so this document does not attempt to tell you how to do this. However, you should probably configure sendmail so that your Internet service provider is designated as your "smart relay" host (the `sendmail.cf` **DS** option). (For more sendmail configuration info, see the sendmail documents - and look at the m4 configurations that come with sendmail. There is almost certain to be one there that will meet your needs).

There are also excellent books on Sendmail (notably the 'bible' from O'Reilly and Associates), but these are almost certainly overkill for most users!

Once you have sendmail configured, you will probably want to have sendmail dispatch any messages that have been sitting in the outbound mail queue as soon as the PPP connection comes up. To do this, add the command

```
sendmail -q &
```

to your /etc/ppp/ip-up script.

Inbound email is a problem for dynamic IP numbers. The way to handle this is to:-

- configure your mail user agent so that all mail is sent out with a "reply to" header giving your email address at your Internet Service provider.
  If you can, you should also set your FROM address to be your email address at your ISP as well.
- use the popclient program to retrieve your email from your service provider.

You can automate this process at dial up time by putting the necessary commands in the `/etc/ppp/ip-up` script.

25.2 Setting Up a local Name server

Whilst you can quite happily use the domain name servers located at your ISP, you can also set up a local caching only (secondary) name server that is brought up by the ip-up script. The advantage of running a local (caching only) name server is that it will save you time (and bandwidth) if you frequently contact the same sites during a long on-line session.

DNS configuration for a caching only nameserver (that uses a "forwarders' line in the named.boot file pointing at your ISPs DNS) is relatively simple. The O'Reilly book (DNS and Bind) explains all you want to know about this.

There is also a DNS-HOWTO available.

One point of Nettiquette: ask permission of your ISP before you start using a secondary, caching only name server in your ISP's domain. Properly configured, your DNS will not cause any problems to your ISP at all, but if you get things wrong, it can cause problems...

26 Setting up a PPP server

As already mentioned, there are many ways to do this. What I present here is the way I do it (using a Cyclades multi-port serial card) and a rotary dial in set of telephone lines.

If you don't like the method I present here, please feel free to go your own way. I would however, be pleased to include additional methods in future versions of the HOWTO. So, please send me your comments and methods!

Please note, this section only concerns setting up Linux as a PPP server. I do not (ever) intend to include information on setting up special terminal servers and such.

Also, I have yet to experiment with shadow passwords (but will be doing so sometime). Information currently presented does NOT therefore include any bells and whistles that are required by the shadow suite.

26.1 Kernel compilation

All the earlier comments regarding kernel compilation and kernel versions versus pppd versions apply. This section assumes that you have read the earlier sections of this document!

For a PPP server, you **MUST** include IP forwarding in your kernel. You may also wish to include other capabilities (such as IP firewalls, accounting etc etc).

If you are using a multi-port serial card, then you must obviously include the necessary drivers in your kernel too!

26.2 Overview of the server system

We offer dial up PPP (and SLIP) accounts and shell accounts using the same username/password pair. This has the advantages (for us) that a user requires only one account and can use it for all types of connectivity.

As we are an educational organisation, we do not charge our staff and students for access, and so do not have to worry about accounting and charging issues.

We operate a firewall between our site and the Internet, and this restricts some user access as the dial up lines are inside our (Internet) firewall (for fairly obvious reasons, details of our other internal firewalls are not presented here and are irrelevant in any case).

The process a user goes through to establish a PPP link to our site (once they have a valid account of course) is :-

- Dial into our rotary dialer (this is a single phone number that connects to a bank of modems - the first free modem is then used).
- Log in using a valid username and password pair.
- At the shell prompt, issue the command `ppp` to start PPP on the server.
- Start PPP on their PC (be it running Windows, DOS, Linux MAC OS or whatever - that is their problem).

The server uses individual `/etc/ppp/options.ttyXX` files for each dial in port that set the remote IP number for dynamic IP allocation. The server users proxyarp routing for the remote clients (set via the appropriate option to pppd). This obviates the need for routed or gated.

When the user hangs up at their end, pppd detects this and tells the modem to hang up, bringing down the PPP link at the same time.

26.3 Getting the software together

You will need the following software:-

- Linux, properly compiled to include the necessary options.
- The appropriate version of pppd for your kernel.
- A 'getty' program that intelligently handles modem communications.
  We use getty_ps2.0.7h, but mgetty is highly thought of. I understand that mgetty can detect a call that is using pap/chap (pap is the standard for Windows95) and invoke pppd automatically, but I have yet to explore this.
- An operational domain name server (DNS) that is accessible to your dial up users.
  You should really be running your own DNS if possible...

26.4 Setting up standard (shell access) dialup.

Before you can set up your PPP server, your Linux box must be capable of handling standard dial up access.

This howto does NOT cover setting this up. Please see the documentation of the getty of your choice and serial HOWTO for information on this.

26.5 Setting up the PPP options files

You will need to set up the overall `/etc/ppp/options` with the common options for all dial up ports. The options we use are:-

```
asyncmap 0
netmask 255.255.254.0
proxyarp
lock
crtscts
modem
```

Note - we do NOT use any (obvious) routing - and in particular there is no defaultroute option. The reason for this is that all you (as a PPP server) are required to do is to route packets **from** the ppp client out across your LAN/Internet and route packets **to** the client from your LAN and beyond.

All that is necessary for this is a host route to the client machine and the use of the 'proxyarp' option to pppd.

The 'proxyarp' option sets up (surprise) a proxy arp entry in the PPP server's arp table that basically says 'send all packets destined for the PPP client to me'. This is the easiest way to set up routing to a single PPP client - but you cannot use this if you are routing between two LANs - you must add proper network routes which can't use proxy arp.

You will almost certainly wish to provide dynamic IP number allocation to your dial up users. You can accomplish this by allocating an IP number to each dial up port. Now, create a `/etc/ppp/options.ttyXX` for each dial up port.

In this, simply put the local (server) IP number and the IP number that is to be used for that port. For example

```
kepler:slip01
```

In particular, note that you can use valid host names in this file (I find that I only remember the IP numbers of critical machines and devices on my networks - names are more meaningful)!

26.6 Setting pppd up to allow users to (successfully) run it

As starting a ppp link implies configuring a kernel device (a network interface) and manipulating the kernel routing tables, special privileges are required - in fact full root privileges.

Fortunately, pppd has been designed to be 'safe' to run set uid to root. So you will need to

```
chmod u+s /usr/sbin/pppd
```

When you list the file, it should then appear as

```
-rwsr-xr-x   1 root     root        74224 Apr 28 07:17 /usr/sbin/pppd
```

If you do not do this, users will be unable to set up their ppp link.

26.7 Setting up the global alias for pppd

In order to simplify things for our dial up PPP users, we create a global alias (in /etc/bashrc) so that one simple command will start ppp on the server once they are logged in.

This looks like

```
alias ppp="exec /usr/sbin/pppd -detach"
```

What this does is

- exec : this means replace the running program (in this case the shell) with the program that is run.
- pppd -detach : start up pppd and do NOT fork into the background. This ensures that when pppd exits there is no process hanging around.

When a user logs in like this, they will appear in the output of 'w' as

```
  6:24pm  up 3 days,  7:00,  4 users,  load average: 0.05, 0.03, 0.00
User     tty       login@  idle   JCPU   PCPU  what
hartr    ttyC0     3:05am  9:14               -
```

And that is it...I told you this was a simple, basic PPP server system!

27 Using PPP across a null modem (direct serial) connection

This is very simple - there is no modem in the way so things are much simpler.

First of all, choose one of the machines as a 'server', setting up a getty on the serial port so you can test that you do have connectivity using minicom to access the serial port on the 'client'.

Once you have this functioning, you can remove the getty UNLESS you want to make sure that the connection is validated using username/password pairs as for a dial up connection. As you have 'physical control' of both machines, I will presume that you do NOT want to do this.

Now, on the server, remove the getty and make sure that you have the serial ports on both machines configured correctly using 'setserial'.

All you need to do now is to start pppd on both systems. I will assume that the connection uses /dev/cua4 on both machines. So, on both machines execute the command:-

```
pppd -detach crtscts lock <local IP>:<remote IP> /dev/cua4 38400 &
```

This will bring up the link - but as yet you have no routing specified. You can test the link by pinging to and fro to each machine. If this works, bring down the link by killing one of the pppd processes.

The routing you need will of course depend on exactly what you are trying to do of course. generally, one of the machines will be connected to an Ethernet (and beyond) and so the routing required is exactly the same as for a PPP server and client.

So on the Ethernet equipped machine, the pppd command would be

```
pppd -detach crtscts lock proxyarp <local IP>:<remote IP> /dev/cua4 38400 &
```

and on the other machine

```
pppd -detach crtscts lock defaultroute <local IP>:<remote IP> /dev/cua4 38400 &
```

If you are linking to networks (using a serial link!) or have more complex routing requirements, you can use /etc/ppp/ip-up in exactly the same way as mentioned earlier in this document.

Robert Hart
Port Hedland, Western Australia
August 1996