1.112.4
Configure Linux as a PPP client
Weight 3

Linux Professional Institute Certification — 102

Nick Urbanik <nicku@nicku.org>
This document Licensed under GPL—see section 16

2005 October
Outline

Context
Objectives
What is PPP?
PPP Clients and Servers
Overview of PPP connection
Establishing the serial connection: chat scripts
An example chat script
Using wvdial to dial
Flow Control
Names of serial ports in Linux systems

Speed of data
pppd options
Where are the options put?
The options
Other methods of authentication
ISDN
ADSL
ip-up, ip-down
Troubleshooting Connections
License Of This Document
1.112.1 Fundamentals of TCP/IP [4]
1.112.3 TCP/IP configuration and troubleshooting [7]
1.112.4 Configure Linux as a PPP client [3]
Candidates should understand the basics of the PPP protocol and be able to configure and use PPP for outbound connections. This objective includes the definition of the chat sequence to connect (given a login example) and the setup commands to be run automatically when a PPP connection is made. It also includes initialisation and termination of a PPP connection, with a modem, ISDN or ADSL and setting PPP to automatically reconnect if disconnected.
Key files, terms, and utilities include:

1.112.4 Configure Linux as a PPP client [3]

/etc/ppp/options.* — main configuration file for pppd

/etc/ppp/peers/* — allow users to use privileged options configured by the administrator

/etc/wvdial.conf — configuration for the wvdial dialer

/etc/ppp/ip-up — script executed when PPP connection made

/etc/ppp/ip-down — script executed when PPP connection terminates

wvdial — a PPP dialer

pppd — The daemon that implements the PPP protocol
What is PPP?

- The Point-to-Point Protocol (PPP) provides a connection between two systems using a serial interface.
- Usually used with ADSL, a dial-up modem, or ISDN modem.
- Can also use over a direct connection called a null modem.
- I have often used PPP over a secure shell connection to create a tunnel over SSH.
- Creates a new network interface: first is called ppp0, next one is ppp1, ... .
- Kernel needs to support PPP, but I have found no recent desktop distributions that do not provide compiled-in support for PPP.
PPP Clients and Servers

- The same software — `pppd` — is used both for PPP dialup clients and PPP servers.
- A server simply uses software like `mgetty` to answer a modem, and an `/etc/initab` entry like:
  
  ```
  S1:2345:respawn:/sbin/mgetty ttyS1
  ```

  ... with some setup in
  
  ```
  /etc/mgetty+sendfax/mgetty.config
  ```

- However, the LPI objectives only include use of Linux as a client.
Overview of PPP connection

1. Client connects to server through a serial connection
2. Client authenticates to the server.
3. PPP starts on the client
4. PPP server usually provides address to client; client and server agree on various other parameters
5. Client configures its network interface using information provided by the server, often including a default route.
Establishing the serial connection: chat scripts

- Much of the work in establishing the PPP connection is getting the modem to dial and talk with the other end.
- We use a program called chat, configured by a chat script for this, and also possibly for logging in, and sometimes starting PPP.
- Questions about chat scripts seem to appear in LPIC exams.
An example chat script

- Consists of pairs of *expected response* then *string sent*

ABORT BUSY
ABORT ERROR
ABORT 'NO CARRIER'
ABORT 'Invalid Login'
ABORT 'Login Incorrect'
'' ATZ
OK ATDT95672314
CONNECT ''
ogin: nicku
ssword: ⟨password⟩
TIMEOUT 5
> ppp
The chat script above was written for a session that looked like this:

```
ATZ
OK
ATDT95672314
CONNECT 31200/ARQ/LAPM/V42BIS
```

User Access Verification

```
login: nicku
Password: ⟨password⟩
```

```
msusw2> ppp
Entering PPP mode.
...
```
Using **wvdial** to dial

- The **wvdial** program helps simplify the connection
- The program **wvdialconf** helps write a configuration file ~/.wvdialrc or /etc/wvdial.conf which may need minor tweaking to use with **pppd**
- You can then use **wvdial** to dial the server, call **pppd** and handle the authentication
- In Ubuntu Hoary, I found this the simplest way to set up the use of PPP.
Flow Control

- A serial interface has a buffer which, when full, cannot accept more information
- Any more information received is lost
- Flow control is a way to stop this happening before the buffer is full
- There are two methods of flow control:
  - **hardware flow control** uses two standard serial control lines: *ready-to-send* (RTS) and *clear-to-send* (CTS)
    - enabled with `pppd` using the option `crtscrts`
  - **software flow control** uses two characters: *XON* and *XOFF* to send flow control information mixed with the data
    - less efficient than hardware flow control
The first serial port is /dev/ttyS0, the second is /dev/ttyS1,...

Ancient text books, or ignorant authors of new books using obsolete sources :-) may refer to serial ports as /dev/cua0, /dev/cua1,...
Speed of data

- Compression allows the flow of data in and out of a modem to exceed the maximum speed of transmission over the phone line by a factor of four
  - so set the serial speed higher than the modem is capable of transmitting
- The speed of data transfer over a serial link is measured in bits per second (bps)
- The term “baud” is not the same as “bits per second”
  - baud is a measure of symbols per second. If there is only one bit per symbol, then baud = bit rate, but usually they are different
  - All the same, much confusion exists, and in wvdial.conf, the speed in bits per second is set using an option “Baud”.
**pppd options**

- **pppd options are set:**
  - in `/etc/ppp/options`
  - in `/etc/ppp/peers/*`
    - Enables ordinary users to use options otherwise disallowed by non-root users
  - on the command line of **pppd**
Frequently used options

asyncmap ⟨map⟩ — specifies which control characters (ASCII characters lower than space) will be sent as a two-byte escape sequence; usually, need this to be set to zero

connect ⟨script⟩ — the script called before PPP protocol starts, usually chat ⟨chat-script⟩ to dial the number

crtscts — Use hardware flow control

ddebug — debugging info to syslog for troubleshooting

defaultroute — set up a default route via remote PPP peer

lock — create a lock file to get exclusive access to the serial device

nodetach — keep pppd in the foreground for debugging

call ⟨name⟩ — read options from /etc/ppp.peers/name, including privileged options

persist — restart the PPP connection if it fails for any reason. *Mentioned in objectives*
Other methods of authentication

The chap example above showed a client logging in responding to username/password prompts (uses `pppd` option `noauth`)

**PAP** — Password Authentication Protocol: started by client sending a username, password. This information is stored in a simple text file, `/etc/ppp/pap-secrets`. I have used PAP with most ISPs.

**CHAP** — Challenge Handshake Authentication Protocol: started by server, which sends its name and a challenge. Client responds with its name and a value derived from the challenge and its authentication information, which is stored in `/etc/ppp/chap-secrets`

**MSCHAP** — It will be a surprise to most that Microsoft, normally a vigorous supporter of IETF and other Internet standards, has gone out on a limb with this one, but it is a Microsoft specific version of CHAP used on MS RAS servers.
Most ISDN modems are configured the same way as other dial-up modems.

I configured a chat script for my parents’ ISDN modem using `wvdial` on Ubuntu Hoary Hedgehog.

Main difference: the device is `/dev/ttyACM0`
I use the package `rp-pppoe` from http://www.roaringpenguin.com/pppoe/ to connect via ADSL to my Internet Service Provider (ISP).

See the documentation with the package.
The script `/etc/ppp/ip-up` is executed when the PPP connection is established.

The script `/etc/ppp/ip-down` is executed when the PPP connection is disconnected.

I have used `/etc/ppp/ip-up.local` (executed from `/etc/ppp/ip-up`) to add static routes when making other PPP connections besides to my ISP.

- Also useful for telling a dynamic DNS provider that your IP address has changed.

Both are called with six parameters to let you know what IP addresses you have, ...
Troubleshooting Connections

- A *terminal program* is your best tool for troubleshooting a serial connection; examples:
  - **minicom** — the program most likely to be asked about in the exam
  - **kermit** — a more powerful terminal program that has finally been included in most modern distributions since the license was made explicitly free for Linux distributors

- Add the option `debug` to PPP options
- Add the option `-v` to `chat`
- Use the option `nodetach` to keep `pppd` in the foreground
- Increase the level of detail to `debug` logged by facility daemon in `/etc/syslog.conf`
- Monitor the `/var/log/messages` log file
Topics Covered

Context
Objectives
What is PPP?
PPP Clients and Servers
Overview of PPP connection
Establishing the serial connection: chat scripts
  An example chat script
  Using wvdial to dial
Flow Control
Names of serial ports in Linux systems
Speed of data
pppd options
  Where are the options put?
  The options
Other methods of authentication
ISDN
ADSL
ip-up, ip-down
Troubleshooting Connections
License Of This Document
Configure Linux as a PPP client

Nick Urbanik

Context

Objectives

What is PPP?

PPP Clients and Servers

Overview of PPP connection

Establishing the serial connection: chat scripts

Flow Control

Names of serial ports in Linux systems

Speed of data

pppd options

Other methods of authentication

ISDN

ADSL

ip-up, ip-down

Troubleshooting Connections

License Of This Document

Copyright © 2005 Nick Urbanik <nicku@nicku.org>
You can redistribute modified or unmodified copies of this document provided that this copyright notice and this permission notice are preserved on all copies under the terms of the GNU General Public License as published by the Free Software Foundation — either version 2 of the License or (at your option) any later version.