

Local GPS NTP Time Server

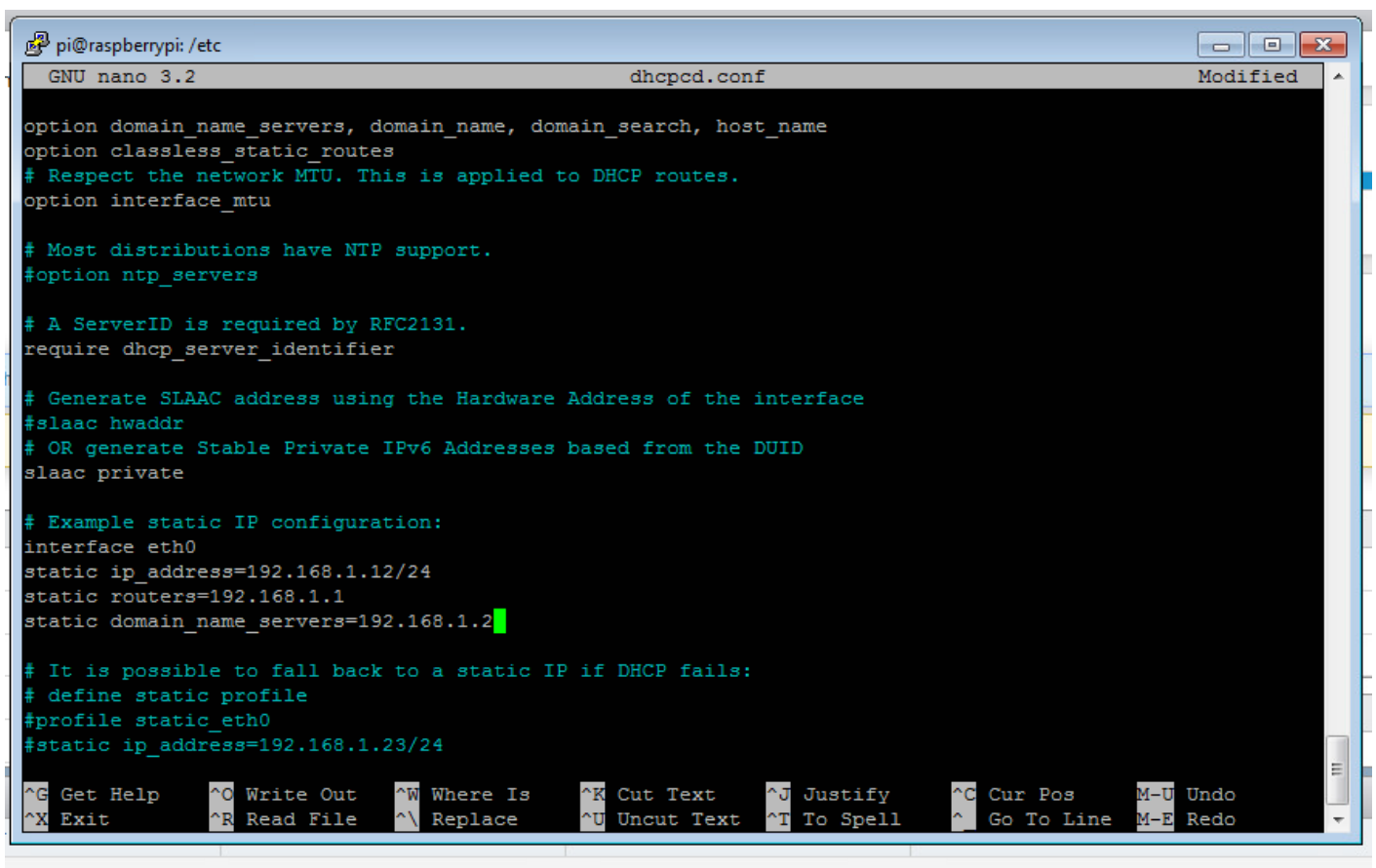
Accessible at: 192.168.1.12

Project files stored here: "\\192.168.1.11\zfs_mirror3\Projects\Dormant\NTP Raspberry Pi Server"

Build Data

Runs Raspian on a Raspberry Pi

Static IP Address was set in: /etc/dhcpd.conf



```
pi@raspberrypi: /etc
GNU nano 3.2          dhcpd.conf          Modified
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# Most distributions have NTP support.
#option ntp_servers

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate SLAAC address using the Hardware Address of the interface
#slaac hwaddr
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
interface eth0
static ip_address=192.168.1.12/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.2

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^ Go To Line    M-E Redo
```

Installed ntpstat with this:

```
sudo apt install ntpstat
```

Usage

Here's some descriptions of the different tools for administrating NTP.

NTPSTAT

Use `ntpstat` to check if the NTP service is in sync:

```
ntpstat
```

It will return something like this:

```
synchronised to NTP server (149.20.54.20) at stratum 3
time correct to within 42 ms
polling server every 1024 s
```

For our local NTP server (with GPS receiver), we see this:

```
pi@raspberrypi:~ $ ntpstat
synchronised to UHF radio at stratum 1
time correct to within 2 ms
polling server every 64 s
```

The `ntpstat` utility also returns an exit status, as a quick tell of synchronization.

Use this, following a call to `ntpstat`:

```
echo $?
```

The return value will mean one of these:

- If exit status 0 - Clock is synchronised.
- exit status 1 - Clock is not synchronised.
- exit status 2 - If clock state is indeterminant, for example if `ntpd` is not contactable.

NTPQ

Use `ntpq -p` to see the following:

```

pi@ntp1pi:~ $ ntpq -p
      remote           refid      st t when poll reach  delay  offset  jitter
=====
0.debian.pool.n .POOL.    16 p   -   64    0    0.000   0.000   0.001
PPS(0)          .PPS.     0 l   -   16    0    0.000   0.000   0.000
*SHM(0)         .GPS.     0 l   9   16   17    0.000  -3.132   4.114
casadeyork.com 163.237.218.19 2 u   25   64    1   28.394 -19.914   0.001
time.cloudflare 10.4.0.197 3 u   22   64    1   18.746 -19.056   0.001
squeakycode.net 128.194.254.9 3 u   23   64    1   69.353 -13.644   0.001
europa.ellipse. 128.252.19.1 2 u   20   64    1   63.242 -21.382   0.001
pi@ntp1pi:~ $

```

PPTSTEST

Use this to test the pps signal:

```
sudo ppstest /dev/pps0
```

The output should spit out a new line every second that looks something like this (your output will be a bit farther from x.000000 since it isn't yet using the GPS PPS):

```

1. pi@pi-ntp:~ $ sudo ppstest /dev/pps0
2. trying PPS source "/dev/pps0"
3. found PPS source "/dev/pps0"
4. ok, found 1 source(s), now start fetching data...
5. source 0 - assert 1618799061.999999504, sequence: 882184 - clear 0.000000000, sequence: 0
6. source 0 - assert 1618799062.999999305, sequence: 882185 - clear 0.000000000, sequence: 0
7. source 0 - assert 1618799063.999997231, sequence: 882186 - clear 0.000000000, sequence: 0
8. source 0 - assert 1618799064.999996827, sequence: 882187 - clear 0.000000000, sequence: 0
9. source 0 - assert 1618799065.999995852, sequence: 882188 - clear 0.000000000, sequence: 0
10. ^C

```

GPSMON

Use the gps monitor function to check the realtime availability of GPS and satellite count and position, with this:

```
gpsmon
```

This will present the following:

References

How to troubleshoot an NTP server: <https://support.ntp.org/Support/TroubleshootingNTP>

The following link is probably not what was used to build the house time server. But, this is an interesting reference on how to build a Raspberry pi GPS time server:

[Building a Raspberry Pi Stratum 1 NTP Server](#)

Here's a good reference on how to tune and configure the NTP setup on a RaspBerry PI: [Building a Raspberry-Pi Stratum-1 NTP Server](#)

Another reference: [Millisecond accurate Chrony NTP with a USB GPS for \\$12 USD - Austin's Nerdy Things](#)

Good reference on the Shared Memory Driver, and how the 127.127.28.x address is used:
<http://doc.ntp.org/archives/drivers/driver28/>

Revision #5

Created 4 September 2025 04:08:59 by glwhite

Updated 4 September 2025 05:21:31 by glwhite