Frequently Asked Questions

Table of Contents

1. chrony compared to other programs

1.1. How does chrony compare to ntpd?

chronyd was designed to work well in a wide range of conditions and it can usually synchronise the system clock faster and with better time accuracy. It doesn't implement some of the less useful NTP modes like broadcast client or multicast server/client.

If your computer is connected to the Internet only for few minutes at a time, the network connection is often congested, you turn your computer off or suspend it frequently, the clock is not very stable (e.g. there are rapid changes in the temperature or it's a virtual machine), or you want to use NTP on an isolated network with no hardware reference clocks in sight, chrony will probably work much better for you.

For a more detailed comparison of features and performance, see the comparison page on the chrony website.

2. Configuration issues

2.1. What is the minimum recommended configuration for an NTP client?

First, the client needs to know which NTP servers it should ask for the current time. They are specified by the server or pool directive. The pool directive can be used for names that resolve to multiple addresses. For good reliability the client should have at least three servers. The iburst option speeds up the initial synchronisation.

To stabilise the initial synchronisation on the next start, the estimated drift of the system clock is saved to a file specified by the driftfile directive.

If the system clock can be far from the true time after boot for any reason, chronyd should be allowed to correct it quickly by stepping instead of slewing, which would take a very long time. The makestep directive does that.

In order to keep the real-time clock (RTC) close to the true time, so the system time is reasonably close to the true time when it's initialised on the

next boot from the RTC, the `rtcsync` directive enables a mode in which the system time is periodically copied to the RTC. It is supported on Linux and macOS.

If you want to use public NTP servers from the pool.ntp.org project, the minimal *chrony.conf* file could be:

```
pool pool.ntp.org iburst
driftfile /var/lib/chrony/drift
makestep 1 3
rtcsync
```

2.2. How do I make an NTP server from an NTP client?

You need to add an `allow` directive to the *chrony.conf* file in order to open the NTP port and allow `chronyd` to reply to client requests. `allow` with no specified subnet allows access from all IPv4 and IPv6 addresses.

2.3. I have several computers on a LAN. Should be all clients of an external server?

The best configuration is usually to make one computer the server, with the others as clients of it. Add a `local` directive to the server's *chrony.conf* file. This configuration will be better because

- the load on the external connection is less
- the load on the external NTP server(s) is less
- if your external connection goes down, the computers on the LAN will maintain a common time with each other.

2.4. Must I specify servers by IP address if DNS is not available on chronyd start?

No. Starting from version 1.25, `chronyd` will keep trying to resolve the names specified by the `server`, `pool`, and `peer` directives in an increasing interval until it succeeds. The `online` command can be issued from `chronyc` to force `chronyd` to try to resolve the names immediately.

2.5. How can I make `chronyd` more secure?

If you don't need to serve time to NTP clients or peers, you can add `port 0` to the *chrony.conf* file to completely disable the NTP server functionality and prevent NTP requests from reaching `chronyd`. Starting from version 2.0, the NTP server port is open only when client access is allowed by the `allow` directive or command, an NTP peer is configured, or the `broadcast` directive is used.

If you don't need to use `chronyc` remotely, you can add the following

directives to the configuration file to bind the command sockets to the loopback interface. This is done by default since version 2.0.

```
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
```

If you don't need to use `chronyc` at all or you need to run `chronyc` only under the root or *chrony* user (which can access `chronyd` through a Unix domain socket since version 2.2), you can disable the internet command sockets completely by adding `cmdport 0` to the configuration file.

You can specify an unprivileged user with the `-u` option, or the `user` directive in the *chrony.conf* file, to which `chronyd` will switch after start in order to drop root privileges. The configure script has a `--with-user` option, which sets the default user. On Linux, `chronyd` needs to be compiled with support for the `libcap` library. On other systems, `chronyd` forks into two processes. The child process retains root privileges, but can only perform a very limited range of privileged system calls on behalf of the parent.

Also, if `chronyd` is compiled with support for the Linux secure computing (seccomp) facility, you can enable a system call filter with the `-F` option. It will significantly reduce the kernel attack surface and possibly prevent kernel exploits from the `chronyd` process if it's compromised. It's recommended to enable the filter only when it's known to work on the version of the system where `chrony` is installed as the filter needs to allow also system calls made from libraries that `chronyd` is using (e.g. libc) and different versions or implementations of the libraries may make different system calls. If the filter is missing some system call, `chronyd` could be killed even in normal operation.

2.6. How can I improve the accuracy of the system clock with NTP sources?

Select NTP servers that are well synchronised, stable and close to your network. It's better to use more than one server, three or four is usually recommended as the minimum, so `chronyd` can detect servers that serve false time and combine measurements from multiple sources.

If you have a network card with hardware timestamping supported on Linux, it can be enabled by the **`hwtimestamp`** directive in the *chrony.conf* file. It should make local receive and transmit timestamps of NTP packets much more accurate.

There are also useful options which can be set in the `server` directive, they are `minpoll`, `maxpoll`, `polltarget`, `maxdelay`, `maxdelayratio`, `maxdelaydevratio`, and `xleave`.

The first three options set the minimum and maximum allowed polling interval, and how should be the actual interval adjusted in the specified range. Their default values are 6 (64 seconds) for `minpoll`, 10 (1024 seconds) for `maxpoll` and 8 (samples) for `polltarget`. The default

values should be used for general servers on the Internet. With your own NTP servers, or if you have permission to poll some servers more frequently, setting these options for shorter polling intervals may significantly improve the accuracy of the system clock.

The optimal polling interval depends mainly on two factors, stability of the network latency and stability of the system clock (which mainly depends on the temperature sensitivity of the crystal oscillator and the maximum rate of the temperature change).

An example of the directive for an NTP server on the Internet that you are allowed to poll frequently could be

```
server foo.example.net minpoll 4 maxpoll 6
polltarget 16
```

An example using very short polling intervals for a server located in the same LAN could be

```
server ntp.local minpoll 2 maxpoll 4 polltarget 30
```

The maxdelay options are useful to ignore measurements with larger delay (e.g. due to congestion in the network) and improve the stability of the synchronisation. The `maxdelaydevratio` option could be added to the example with local NTP server

```
server ntp.local minpoll 2 maxpoll 4 polltarget 30
maxdelaydevratio 2
```

If your server supports the interleaved mode, the `xleave` option should be added to the `server` directive in order to allow the server to send the client more accurate hardware or kernel transmit timestamps. When combined with local hardware timestamping, sub-microsecond accuracy may be possible. An example could be

```
server ntp.local minpoll 2 maxpoll 2 xleave
hwtimestamp eth0
```

2.7. Does **chronyd** have an ntpdate mode?

Yes. With the `-q` option `chronyd` will set the system clock once and exit. With the `-Q` option it will print the measured offset without setting the clock. If you don't want to use a configuration file, NTP servers can be specified on the command line. For example:

```
# chronyd -q 'pool pool.ntp.org iburst'
```

2.8. What happened to the `commandkey` and `generatecommandkey` directives?

They were removed in version 2.2. Authentication is no longer supported in the command protocol. Commands that required authentication are now allowed only through a Unix domain socket, which is accessible only by the root and *chrony* users. If you need to configure `chronyd` remotely or locally without the root password, please consider using ssh and/or sudo to run `chronyc` under the root or *chrony* user on the host where `chronyd` is running.

3. Computer is not synchronising

This is the most common problem. There are a number of reasons, see the following questions.

3.1. Behind a firewall?

Check the `Reach` value printed by the `chronyc`'s `sources` command. If it's zero, it means `chronyd` did not get any valid responses from the NTP server you are trying to use. If there is a firewall between you and the server, the packets may be blocked. Try using a tool like `wireshark` or `tcpdump` to see if you're getting any responses from the server.

When `chronyd` is receiving responses from the servers, the output of the `sources` command issued few minutes after `chronyd` start might look like this:

```
210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx
Last sample
===============================================================

^* foo.example.net               2   6   377    34
+484us[ -157us] +/-   30ms
^- bar.example.net               2   6   377    34
+33ms[  +32ms] +/-   47ms
^+ baz.example.net               3   6   377    35
-1397us[-2033us] +/-   60ms
```

3.2. Are NTP servers specified with the `offline` option?

Check that you're using `chronyc`'s `online` and `offline` commands appropriately. The `activity` command prints the number of sources that are currently online and offline. For example:

```
200 OK
3 sources online
0 sources offline
0 sources doing burst (return to online)
```

```
0 sources doing burst (return to offline)
0 sources with unknown address
```

3.3. Is chronyd allowed to step the system clock?

By default, chronyd adjusts the clock gradually by slowing it down or speeding it up. If the clock is too far from the true time, it will take a long time to correct the error. The System time value printed by the chronyc's tracking command is the remaining correction that needs to be applied to the system clock.

The makestep directive can be used to allow chronyd to step the clock. For example, if *chrony.conf* had

```
makestep 1 3
```

the clock would be stepped in the first three updates if its offset was larger than one second. Normally, it's recommended to allow the step only in the first few updates, but in some cases (e.g. a computer without an RTC or virtual machine which can be suspended and resumed with an incorrect time) it may be necessary to allow the step on any clock update. The example above would change to

```
makestep 1 -1
```

3.4. Using a Windows NTP server?

A common issue with Windows NTP servers is that they report a very large root dispersion (e.g. three seconds or more), which causes chronyd to ignore the server for being too inaccurate. The sources command may show a valid measurement, but the server is not selected for synchronisation. You can check the root dispersion of the server with the chronyc's ntpdata command.

The maxdistance value needs to be increased in *chrony.conf* to enable synchronisation to such a server. For example:

```
maxdistance 16.0
```

3.5. Using a PPS reference clock?

A pulse-per-second (PPS) reference clock requires a non-PPS time source to determine which second of UTC corresponds to each pulse. If it is another reference clock specified with the lock option in the refclock directive, the offset between the two reference clocks must be smaller than 0.2 seconds in order for the PPS reference clock to work. With NMEA

reference clocks it is common to have a larger offset. It needs to be corrected with the `offset` option.

One approach to find out a good value of the `offset` option is to configure the reference clocks with the `noselect` option and compare them to an NTP server. For example, if the `sourcestats` command showed

```
Name/IP Address            NP  NR  Span  Frequency
Freq Skew  Offset  Std Dev
==============================================================

PPS0                        0   0     0     +0.000
2000.000     +0ns  4000ms
NMEA                       58  30   231    -96.494
38.406   +504ms  6080us
foo.example.net             7   3   200     -2.991
16.141   -107us   492us
```

the offset of the NMEA source would need to be increased by about 0.504 seconds. It does not have to be very accurate. As long as the offset of the NMEA reference clock stays below 0.2 seconds, the PPS reference clock should be able to determine the seconds corresponding to the pulses and allow the samples to be used for synchronisation.

4. Issues with `chronyc`

4.1. I keep getting the error `506 Cannot talk to daemon`

When accessing `chronyd` remotely, make sure that the *chrony.conf* file (on the computer where `chronyd` is running) has a `cmdallow` entry for the computer you are running `chronyc` on and an appropriate `bindcmdaddress` directive. This isn't necessary for localhost.

Perhaps `chronyd` is not running. Try using the `ps` command (e.g. on Linux, `ps -auxw`) to see if it's running. Or try `netstat -a` and see if the ports 123/udp and 323/udp are listening. If `chronyd` is not running, you may have a problem with the way you are trying to start it (e.g. at boot time).

Perhaps you have a firewall set up in a way that blocks packets on port 323/udp. You need to amend the firewall configuration in this case.

4.2. I keep getting the error `501 Not authorised`

Since version 2.2, the `password` command doesn't do anything and `chronyc` needs to run locally under the root or *chrony* user, which are allowed to access the `chronyd`'s Unix domain command socket.

With older versions, you need to authenticate with the `password`

command first or use the `-a` option to authenticate automatically on start. The configuration file needs to specify a file which contains keys (`keyfile` directive) and which key in the key file should be used for `chronyc` authentication (`commandkey` directive).

4.3. Why does `chronyc tracking` always print an IPv4 address as reference ID?

The reference ID is a 32-bit value and in versions before 3.0 it was printed in quad-dotted notation, even if the reference source did not actually have an IPv4 address. For IPv4 addresses, the reference ID is equal to the address, but for IPv6 addresses it is the first 32 bits of the MD5 sum of the address. For reference clocks, the reference ID is the value specified with the `refid` option in the `refclock` directive.

Since version 3.0, the reference ID is printed as a hexadecimal number to avoid confusion with IPv4 addresses.

If you need to get the IP address of the current reference source, use the `-n` option to disable resolving of IP addresses and read the second field (printed in parentheses) on the `Reference ID` line.

4.4. Is the `chronyc` / `chronyd` protocol documented anywhere?

Only by the source code. See *cmdmon.c* (`chronyd` side) and *client.c* (`chronyc` side).

5. Real-time clock issues

5.1. What is the real-time clock (RTC)?

This is the clock which keeps the time even when your computer is turned off. It is used to initialise the system clock on boot. It normally doesn't drift more than few seconds per day.

There are two approaches how `chronyd` can work with it. One is to use the `rtcsync` directive, which tells `chronyd` to enable a kernel mode which sets the RTC from the system clock every 11 minutes. `chronyd` itself won't touch the RTC. If the computer is not turned off for a long time, the RTC should still be close to the true time when the system clock will be initialised from it on the next boot.

The other option is to use the `rtcfile` directive, which tells `chronyd` to monitor the rate at which the RTC gains or loses time. When `chronyd` is started with the `-s` option on the next boot, it will set the system time from the RTC and also compensate for the drift it has measured previously. The `rtcautotrim` directive can be used to keep the RTC close to the true time, but it's not strictly necessary if its only purpose is to set

the system clock when `chronyd` is started on boot. See the documentation for details.

5.2. I want to use `chronyd`'s RTC support. Must I disable `hwclock`?

The `hwclock` program is often set-up by default in the boot and shutdown scripts with many Linux installations. With the kernel RTC synchronisation (`rtcsync` directive), the RTC will be set also every 11 minutes as long as the system clock is synchronised. If you want to use `chronyd`'s RTC monitoring (`rtcfile` directive), it's important to disable `hwclock` in the shutdown procedure. If you don't, it will over-write the RTC with a new value, unknown to `chronyd`. At the next reboot, `chronyd` started with the `-s` option will compensate this (wrong) time with its estimate of how far the RTC has drifted whilst the power was off, giving a meaningless initial system time.

There is no need to remove `hwclock` from the boot process, as long as `chronyd` is started after it has run.

5.3. I just keep getting the `513 RTC driver not running` message

For the real-time clock support to work, you need the following three things

- an RTC in your computer
- a Linux kernel with enabled RTC support
- an `rtcfile` directive in your *chrony.conf* file

5.4. I get `Could not open /dev/rtc, Device or resource busy` in my syslog file

Some other program running on the system may be using the device.

5.5. What if my computer does not have an RTC or backup battery?

In this case you can still use the `-s` option to set the system clock to the last modification time of the drift file, which should correspond to the system time when `chronyd` was previously stopped. The initial system time will be increasing across reboots and applications started after `chronyd` will not observe backward steps.

6. NTP-specific issues

6.1. Can chronyd be driven from broadcast/multicast NTP servers?

No, the broadcast/multicast client mode is not supported and there is currently no plan to implement it. While the mode may be useful to simplify configuration of clients in large networks, it is inherently less accurate and less secure (even with authentication) than the ordinary client/server mode.

When configuring a large number of clients in a network, it is recommended to use the `pool` directive with a DNS name which resolves to addresses of multiple NTP servers. The clients will automatically replace the servers when they become unreachable, or otherwise unsuitable for synchronisation, with new servers from the pool.

Even with very modest hardware, an NTP server can serve time to hundreds of thousands of clients using the ordinary client/server mode.

6.2. Can chronyd transmit broadcast NTP packets?

Yes, the `broadcast` directive can be used to enable the broadcast server mode to serve time to clients in the network which support the broadcast client mode (it's not supported in `chronyd`, see the previous question).

6.3. Can chronyd keep the system clock a fixed offset away from real time?

Yes. Starting from version 3.0, an offset can be specified by the `offset` option for all time sources in the *chrony.conf* file.

6.4. What happens if the network connection is dropped without using chronyc's offline command first?

`chronyd` will keep trying to access the sources that it thinks are online, and it will take longer before new measurements are actually made and the clock is corrected when the network is connected again. If the sources were set to offline, `chronyd` would make new measurements immediately after issuing the `online` command.

Unless the network connection lasts only few minutes (less than the maximum polling interval), the delay is usually not a problem, and it may be acceptable to keep all sources online all the time.

7. Operating systems

7.1. Does chrony support Windows?

No. The `chronyc` program (the command-line client used for configuring `chronyd` while it is running) has been successfully built and run under Cygwin in the past. `chronyd` is not portable, because part of it is very system-dependent. It needs adapting to work with Windows' equivalent of the adjtimex() call, and it needs to be made to work as a service.

7.2. Are there any plans to support Windows?

We have no plans to do this. Anyone is welcome to pick this work up and contribute it back to the project.

Last updated 2018-05-25 10:53:21 CEST