NAME

ntpd − Network Time Protocol service daemon

SYNOPSIS

ntpd
[−46aghLmnNqx] [assert] [−c *conffile*] [−f *driftfile*]
[−i *jaildir*] [−k *keyfile*] [−l *logfile*] [−p *pidfile*]
[−P *priority*] [−s *statsdir*] [−t *key*]
[−u *user*[:'group']] [−U *interface_update_interval*]
[−v *variable*] [−V *variable*] [server...]

DESCRIPTION

The ntpd utility is an operating system daemon which sets and maintains the system time of day in synchronism with Internet standard time servers. It is a complete implementation of the Network Time Protocol (NTP) version 4, as defined by RFC 5905, but also retains compatibility with version 3, as defined by RFC 1305, and versions 1 and 2, as defined by RFC 1059 and RFC 1119, respectively.

The ntpd utility can synchronize time to a theoretical precision of about 232 picoseconds. In practice, this limit is unattainable due to quantum limits on the clock speed of ballistic−electron logic.

Ordinarily, ntpd reads the ntp.conf(5) configuration file at startup time in order to determine the synchronization sources and operating modes. It is also possible to specify a working, although limited, configuration entirely on the command line, obviating the need for a configuration file.

The ntpd program normally operates continuously while adjusting the system time and frequency, but in some cases this might not be practical. With the −q option ntpd operates as in continuous mode, but exits just after setting the clock for the first time. Most applications will probably want to specify the iburst option with the server command. With this option, a volley of messages is exchanged to groom the data and set the clock in about ten seconds. With −q, if nothing is heard after a few minutes, the daemon times out and exits without setting the clock.

Various internal ntpd variables can be displayed and configuration options altered while the ntpd is running using the ntpq(1) utility program. The state of ntpd can be continuously monitored using ntpmon(1).

When ntpd starts it looks at the value of umask(2), and if zero ntpd will set the umask(2) to 022.

OPTIONS

−4, −−ipv4
Force IPv4 DNS name resolution. This option must not appear in combination with any of the following options: ipv6.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, −−ipv6
Force IPv6 DNS name resolution. This option must not appear in combination with any of the following options: ipv4.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−a, −−assert
REQUIRE(false) to test assert handler.

−c *string*, −−configfile=*string*
configuration file name.

The name and path of the configuration file, /etc/ntpsec/ntp.conf by default.

−d, −−debug−level
Increase debug verbosity level. This option may appear an unlimited number of times.

−D *number*, −−set−debug−level=*number*
Set the debug verbosity level. This option may appear an unlimited number of times. This option takes an integer number as its argument.

−f *string*, −−driftfile=*string*
frequency drift file name.

The name and path of the frequency file, e.g. /var/lib/ntpsec/ntp.drift. This is the same operation as the *driftfile* configuration specification in the /etc/ntpsec/ntp.conf file.

−g, −−panicgate
Allow the first adjustment to be big. This option may appear an unlimited number of times.

Normally, ntpd exits with a message to the system log if the offset exceeds the panic threshold, which is 1000 s by default. This option allows the time to be set to any value without restriction; however, this can happen only once. If the threshold is exceeded after that, ntpd will exit with a message to the system log. This option can be used with the −q and −x options. See the *tinker* configuration file directive for other options.

−G, −−force−step−once
Step any initial offset correction.

Normally, ntpd steps the time if the time offset exceeds the step threshold, which is 128 ms by default, and otherwise slews the time. This option forces the initial offset correction to be stepped, so the highest time accuracy can be achieved quickly. However, this may also cause the time to be stepped back so this option must not be used if applications requiring monotonic time are running. See the *tinker* configuration file directive for other options.

−h, −−help
Print a usage message summarizing options and exit.

−i *string*, −−jaildir=*string*
Jail directory.

Chroot the server to the directory *jaildir* This option also implies that the server attempts to drop root privileges at startup. You may need to also specify a −u option. This option is only available if the OS supports adjusting the clock without full root privileges. This option is supported under Linux, NetBSD, and Solaris.

−I *iface*, −−interface=*iface*
Listen on an interface name or address. This option may appear an unlimited number of times.

Open the network address given, or all the addresses associated with the given interface name. This option may appear multiple times. This option also implies not opening other addresses, except wildcard and localhost. This option is deprecated. Please consider using the configuration file *interface* command, which is more versatile.

−k *string*, −−keyfile=*string*
the path to symmetric keys.

Specify the name and path of the symmetric key file. /etc/ntpsec/ntp.keys is the default location. This is the same operation as the *keys* configuration file directive.

−l *string*, −−logfile=*string*
the path to the log file.

Specify the name and path of the log file. The default is the system log file. This is the same operation as the *logfile* configuration file directive. See ntp.conf(5) for more info.

−L, −−novirtualips

Do not listen to virtual interfaces.

Do not listen to virtual interfaces, defined as those with names containing a colon. This option is deprecated. Please consider using the configuration file interface command, which is more versatile.

−m, −−mdns
Register with mDNS as an NTP server.

Registers as an NTP server with the local mDNS server which allows the server to be discovered via mDNS client lookup.

−n, −−nofork
Do not fork. This option must not appear in combination with any of the following options: wait−sync.

−N, −−nice
Run at high priority.

To the extent permitted by the operating system, run ntpd at the highest priority.

−p *string*, −−pidfile=*string*
the path to the PID file.

Specify the name and path of the file used to record ntpd's process ID. This is the same operation as the *pidfile* configuration file directive.

−P *number*, −−priority=*number*
Process priority. This option takes an integer number as its argument.

To the extent permitted by the operating system, run ntpd at the specified *sched_setscheduler(SCHED_FIFO)* priority.

−q, −−quit
Set the time and quit. This option must not appear in combination with wait−sync.

ntpd will not daemonize and will exit after the clock is first synchronized. This behavior mimics that of the old *ntpdate* program, which has been replaced with a shell script. The −g and −x options can be used with this option. Note: The kernel time discipline is disabled with this option.

−s *string*, −−statsdir=*string*
Statistics file location.

Specify the directory path for files created by the statistics facility. This is the same operation as the *statsdir* configuration file directive.

−t *tkey*, −−trustedkey=*tkey*
Trusted key number. This option may appear an unlimited number of times.

Add the specified key number to the trusted key list.

−u *string*, −−user=*string*
Run as userid (or userid:groupid).

Specify a user, and optionally a group, to switch to. The user and group may be specified by name or numeric id. If no group is specified, then the default group for userid is used. This option is only available if the OS supports adjusting the clock without full root privileges. This option is supported under Linux, NetBSD, Solaris and other OS.

−U *number*, −−updateinterval=*number*
interval in seconds between scans for new or dropped interfaces. This option takes an integer number as its argument.

Give the time in seconds between two scans for new or dropped interfaces. For systems with routing socket support, the scans will be performed shortly after the interface change has been detected by the system. Use 0 to disable scanning. 60 seconds is the minimum time between scans.

−w *number*, −−wait−sync=*number*
Seconds to wait for first clock sync. This option must not appear in combination with any of the following options: nofork, quit. This option takes an integer number as its argument.

If greater than zero alters ntpd's behavior when forking to daemonize. Instead of exiting with status 0 immediately after the fork, the parent waits up to the specified number of seconds for the child to first synchronize the clock. The exit status is zero (success) if the clock was synchronized; otherwise, it is ETIMEDOUT. This provides the option for a script starting ntpd to easily wait for the first set of the clock before proceeding.

−x, −−slew
Slew up to 600 seconds.

Normally, the time is slewed if the offset is less than the step threshold, which is 128 ms by default, and stepped if above the threshold. This option sets the threshold to 600 s, which is well within the accuracy window to set the clock manually. Note: Since the slew rate of typical Unix kernels is limited to 0.5 ms/s, each second of adjustment requires an amortization interval of 2000 s. Thus, an adjustment as much as 600 s will take almost 14 days to complete. This option can be used with the −g and −q options. See the *tinker* configuration file directive for other options. Note: The kernel time discipline is disabled with this option.

−z *nvar*, −−var=*nvar*
make ARG an ntp variable (RW). This option may appear an unlimited number of times.

−Z *nvar*, −−dvar=*ndvar*
make ARG an ntp variable (RW|DEF). This option may appear an unlimited number of times.

−V, −−version
Output version of program and exit.

Any arguments given after options are interpreted as server addresses or hostnames, with the *iburst* option implied. Associations with these are formed before any associations implied by the configuration file.

USAGE

How NTP Operates

The ntpd utility operates by exchanging messages with one or more configured servers over a range of designated poll intervals. When started, whether for the first or subsequent times, the program requires several exchanges from the majority of these servers so the signal processing and mitigation algorithms can accumulate and groom the data and set the clock. In order to protect the network from bursts, the initial poll interval for each server is delayed an interval randomized over a few seconds. At the default initial poll interval of 64s, several minutes can elapse before the clock is set. This initial delay to set the clock can be safely and dramatically reduced using the *iburst* keyword with the *server* configuration command, as described in ntp.conf(5).

Most operating systems and hardware of today incorporate a time−of−year (TOY) chip to maintain the time during periods when the power is off. When the machine is booted, the chip is used to initialize the operating system time. After the machine has synchronized to an NTP server, the operating system corrects the chip from time to time. In the default case, if ntpd detects that the time on the host is more than 1000s from the server time, ntpd assumes something must be terribly wrong, and the only reliable action is for the operator to intervene and set the clock by hand. (Reasons for this include there is no TOY chip, or its battery is dead, or that the TOY chip is just of poor quality.) This causes ntpd to exit with a panic message to the system log. The −g option overrides this check, and the clock will be set to the server time regardless of the chip time (up to 68 years in the past or future — this is a limitation of the NTPv4 protocol). However, and to protect against broken hardware, such as when the CMOS battery fails or the clock

counter becomes defective, once the clock has been set an error greater than 1000s will cause ntpd to exit anyway.

Under ordinary conditions, ntpd adjusts the clock in small steps so that the timescale is effectively continuous and without discontinuities. Under conditions of extreme network congestion, the roundtrip delay jitter can exceed three seconds and the synchronization distance, which is equal to one−half the roundtrip delay plus error budget terms, can become very large. The ntpd algorithms discard sample offsets exceeding 128 ms, unless the interval during which no sample offset is less than 128 ms exceeds 900s. The first sample after that, no matter what the offset, steps the clock to the indicated time. In practice, this reduces the false alarm rate where the clock is stepped in error to a vanishingly low incidence.

As the result of this behavior, once the clock has been set it very rarely strays more than 128 ms even under extreme cases of network path congestion and jitter. Sometimes, in particular, when ntpd is first started without a valid drift file on a system with a large intrinsic drift the error might grow to exceed 128 ms, which would cause the clock to be set backwards if the local clock time is more than 128 ms in the future relative to the server. In some applications, this behavior may be unacceptable. There are several solutions, however. If the −x option is included on the command line, the clock will never be stepped and only slew corrections will be used. But this choice comes at a cost that should be carefully explored before deciding to use the −x option. The maximum slew rate possible is limited to 500 parts−per−million (PPM) as a consequence of the correctness principles on which the NTP protocol and algorithm design are based. As a result, the local clock can take a long time to converge to an acceptable offset, about 2,000 s for each second the clock is outside the acceptable range. During this interval, the local clock will not be consistent with any other network clock and the system cannot be used for distributed applications that require correctly synchronized network time.

In spite of the above precautions, sometimes when large frequency errors are present the resulting time offsets stray outside the 128−ms range and an eventual step or slew time correction is required. If following such a correction the frequency error is so large that the first sample is outside the acceptable range, ntpd enters the same state as when the *ntp.drift* file is not present. The intent of this behavior is to quickly correct the frequency and restore operation to the normal tracking mode. In the most extreme cases, there may be occasional step/slew corrections and subsequent frequency corrections. It helps in these cases to use the *burst* keyword when configuring the server, but ONLY when you have permission to do so from the owner of the target host.

Finally, in the past, many startup scripts would run a separate utility to get the system clock close to correct before starting ntpd(8), but this was never more than a mediocre hack and is no longer needed. If you are following the instructions in the section called “Starting NTP (Best Current Practice)” and you still need to set the system time before starting ntpd, please open a bug report and document what is going on, and then look at using ntpdig(1).

There is a way to start ntpd(8) that often addresses all of the problems mentioned above.

Starting NTP (Best Current Practice)

First, use the *iburst* option on your *server* and *pool* entries.

If you can also keep a good *ntp.drift* file then ntpd(8) will effectively "warm−start" and your system’s clock will be stable in under 11 seconds' time.

As soon as possible in the startup sequence, start ntpd(8) with at least the −g and perhaps the −N options. Then, start the rest of your "normal" processes. This will give ntpd(8) as much time as possible to get the system’s clock synchronized and stable.

Finally, if you have processes like *dovecot* or database servers that require monotonically−increasing time, run ntpwait(8) as late as possible in the boot sequence (perhaps with the −v flag) and after ntpwait(8) exits successfully it is as safe as it will ever be to start any processes that require stable time.

Frequency Discipline

The ntpd behavior at startup depends on whether the frequency file, usually *ntp.drift*, exists. This file contains the latest estimate of clock frequency error. When the ntpd is started and the file does not exist, the ntpd enters a special mode designed to quickly adapt to the particular system clock oscillator time and frequency error. This takes approximately 15 minutes, after which the time and frequency are set to nominal values and the ntpd enters normal mode, where the time and frequency are continuously tracked relative to the server. After one hour the frequency file is created and the current frequency offset written to it. When the ntpd is started and the file does exist, the ntpd frequency is initialized from the file and enters normal mode immediately. After that, the current frequency offset is written to the file at hourly intervals.

Operating Modes

ntpd normally operates continuously while monitoring for small changes in frequency and trimming the clock for the ultimate precision. However, it can operate in a one−time mode where the time is set from an external server and frequency is set from a previously recorded frequency file.

By default, ntpd runs in continuous mode where each of possibly several external servers is polled at intervals determined by an intricate state machine. The state machine measures the incidental roundtrip delay jitter and oscillator frequency wander and determines the best poll interval using a heuristic algorithm. Ordinarily, and in most operating environments, the state machine will start with 64s intervals and eventually increase in steps to 1024s. A small amount of random variation is introduced in order to avoid bunching at the servers. In addition, should a server become unreachable for some time, the poll interval is increased in steps to 1024s in order to reduce network overhead.

In some cases, it may not be practical for ntpd to run continuously. The −q option is provided to support running ntpd periodically from a cron(8) job. Setting this option will cause ntpd to exit just after setting the clock for the first time. The procedure for initially setting the clock is the same as in continuous mode; most applications will probably want to specify the *iburst* keyword with the *server* configuration command. With this keyword, a volley of messages are exchanged to groom the data and the clock is set in about 10 sec. If nothing is heard after a couple of minutes, the daemon times out and exits.

When kernel support is available to discipline the clock frequency, which is the case for stock Solaris, Linux, and FreeBSD, a useful feature is available to discipline the clock frequency. First, ntpd is run in continuous mode with selected servers in order to measure and record the intrinsic clock frequency offset in the frequency file. It may take some hours for the frequency and offset to settle down. Then the ntpd is stopped and run in one−time mode as required. At each startup, the frequency is read from the file and initializes the kernel frequency.

Poll Interval Control

This version of NTP includes an intricate state machine to reduce the network load while maintaining a quality of synchronization consistent with the observed jitter and wander. There are a number of ways to tailor the operation in order enhance accuracy by reducing the interval or to reduce network overhead by increasing it. However, the user is advised to carefully consider the consequences of changing the poll adjustment range from the default minimum of 64 s to the default maximum of 1,024 s. The default minimum can be changed with the *tinker minpoll* command to a value not less than 16 s. This value is used for all configured associations, unless overridden by the *minpoll* option on the configuration command. Note that most device drivers will not operate properly if the poll interval is less than 64 s and that the broadcast server and manycast client associations will also use the default unless overridden.

In some cases involving dial up or toll services, it may be useful to increase the minimum interval to a few tens of minutes and maximum interval to a day or so. Under normal operation conditions, once the clock discipline loop has stabilized the interval will be increased in steps from the minimum to the maximum. However, this assumes the intrinsic clock frequency error is small enough for the discipline loop correct it. The capture range of the loop is 500 PPM at an interval of 64s decreasing by a factor of two for each doubling of the interval. At a minimum of 1,024 s, for example, the capture range is only 31 PPM. If the intrinsic error is greater than this, the drift file *ntp.drift* will have to be specially tailored to reduce the residual error below this limit. Once this is done, the drift file is automatically updated once per hour and is

available to initialize the frequency on subsequent daemon restarts.

The huff−n'−puff Filter

In scenarios where a considerable amount of data are to be downloaded or uploaded over telephone modems, timekeeping quality can be seriously degraded. This occurs because the differential delays on the two directions of transmission can be quite large. In many cases, the apparent time errors are so large as to exceed the step threshold and a step correction can occur during and after the data transfer is in progress.

The huff−n'−puff filter is designed to correct the apparent time offset in these cases. It depends on knowledge of the propagation delay when no other traffic is present. In common scenarios, this occurs during other than work hours. The filter maintains a shift register that remembers the minimum delay over the most recent interval measured usually in hours. Under conditions of severe delay, the filter corrects the apparent offset using the sign of the offset and the difference between the apparent delay and minimum delay. The name of the filter reflects the negative (huff) and positive (puff) correction, which depends on the sign of the offset.

The filter is activated by the *tinker* command and *huffpuff* keyword, as described in ntp.conf(5).

FILES

File	Default	Option	Option
configuration file	/etc/ntpsec/ntp.conf	−c	conffile
configuration directory	/etc/ntpsec/ntp.d	−c	conffile
frequency file	none	−f	driftfile
leapseconds file	none		leapfile
process ID file	none	−p	pidfile
log file	system log	−l	logfile
include file	none	none	includefile
statistics path	/var/log/ntpsec	−s	statsdir
keys file	none	−k	keys

Configuration files are parsed according to the following rules:

1. The plain config file (normally /etc/ntpsec/ntp.conf but the path can be overridden by the −c option) is read first if it exists.
2. Then the configuration directory, if it exists, is scanned. Normally this directory is /etc/ntpsec/ntp.d, but if the −c option is specified the /etc/ntpsec will be specified by the directory name of the −c argument.
3. Each file beneath the configuration directory with the extension ".conf" is interpreted. Files are interpreted in ASCII sort order of their pathnames. Files with other extensions or no extensions are ignored.

SIGNALS

SIGQUIT, SIGINT, and SIGTERM will cause ntpd to clean up and exit.

SIGHUP will reopen the log file if it has changed and check for a new leapseconds file if one was specified. If the NTS server is enabled, it will reload the certificate file if it has changed.

On most systems, you can send SIGHUP to ntpd with

```
# sigkill −HUP ntpd
```

If built with debugging enabled (waf configured with −−enable−debug) SIGUSR1 will increase the debug level by 1 and SIGUSR2 will decrease it by 1. This may be helpful if you are running with −n, either just to see the logging on your screen or with gdb.

BUGS

The *−V* option is not backward−compatible with its use (as the equivalent of −Z) in older versions.

STANDARDS

RFC 1059
: David L. Mills, *Network Time Protocol (Version 1)*, RFC 1059

RFC 1119
: David L. Mills, *Network Time Protocol (Version 2)*, RFC 1119

RFC 1305
: David L. Mills, *Network Time Protocol (Version 3)*, RFC 1305

RFC 5905
: David L. Mills and J. Martin, Ed. and J. Burbank and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC 5905

RFC 5907
: H. Gerstung and C. Elliott and B. Haberman, Ed., *Definitions of Managed Objects for Network Time Protocol Version 4: (NTPv4)*, RFC 5907

RFC 5908
: R. Gayraud and B. Lourdelet, *Network Time Protocol (NTP) Server Option for DHCPv6*, RFC 5908

EXIT STATUS

One of the following exit values will be returned:

0 (EXIT_SUCCESS)
: Successful program execution.

1 (EXIT_FAILURE)
: Execution failed − examine system logfiles.

SEE ALSO

ntp.conf(5), ntpq(1), ntpdig(1).