NAME

vnstatd – daemon based database updating for vnStat

SYNOPSIS

vnstatd [**–Ddnpstv?**] [**––alwaysadd** [*mode*]] [**––config** *file*] [**––daemon**] [**––debug**] [**–g** *group*] [**––group** *group*] [**––help**] [**––initdb**] [**––noadd**] [**––nodaemon**] [**––noremove**] [**––pidfile** *file*] [**––startempty**] [**––sync**] [**––timestamp**] [**––u** *user*] [**––user** *user*] [**––version**]

DESCRIPTION

The purpose of **vnstatd** is to provide a flexible and robust way for gathering data to the database that **vnstat**(1) uses. The availability of each interface is automatically tracked which removes the need for additional scripts to be implemented and called when an interface comes online or goes offline.

vnstatd is the command for starting the daemon. The daemon can either fork itself to run as a background process or stay attached to the terminal. It supports logging directly to terminal, to a user selectable file or using syslog.

Once started, the daemon will read **vnstat.conf**(5) if available and then check if there is a database present in the database directory that has been specified in the configuration file. By default, if no database is found, a database will be created during startup with entries for all available interfaces excluding pseudo interfaces lo, lo0 and sit0. This automatic database entry creation behaviour can be disabled using the **--noadd** option. Alternatively, the **--alwaysadd** option can be used to instruct the daemon to create new database entries whenever interfaces not currently in the databases become visible. By default, unless the **--startempty** option is used, the daemon will not stay running if no interfaces are discovered during startup and the database contains no interfaces.

The daemon will proceed to track the availability of monitored interfaces, process the interface traffic statistics and write new values to the database at a configured interval. As a result, the daemon ends up spending most of the time sleeping between updates. New interfaces added to the database will be automatically picked up for monitoring without the daemon needing to be notified.

When the **UseUTC** configuration option isn't enabled, data is stored in the database using local time based on the daemon's execution environment when the configuration option isn't enabled. Any changes in the system clock or the system timezone configuration will result in data being inserted according to the new local time without any recalculation being done for already stored data. The daemon and the database in essence aren't aware of the used timezone or possible daylight saving time and cannot be configured to offset the timestamps to any direction. If a system clock or system timezone change or daylight saving time observation ending results in an already seen time period to repeat then the existing database values get incremented with the new data.

OPTIONS

--alwaysadd *[mode]*

Enable automatic creation of new database entries for interfaces not currently in the database even if the database file already exists when the daemon is started. New database entries will also get created for new interfaces seen while the daemon is running. Pseudo interfaces lo, lo0 and sit0 are always excluded from getting added. Using the option without *mode* defined or with *mode* set to 1 will enable the feature. Setting *mode* to 0 will disable the feature. This command line option overrides the **AlwaysAddNewInterfaces** configuration option when used.

--config *file*

Use *file* as configuration file instead of using automatic configuration file search functionality.

-d, --daemon

Fork process to background and run as a daemon.

-D, --debug
Provide additional output for debug purposes. The process will stay attached to the terminal for output.

-g, --group *group*
Set daemon process group to *group* during startup. *group* can be either the name of the group or a numerical group id. This option can only be used when the process is started as root.

--initdb
Create a new database, import data from found legacy databases if **--noadd** option isn't used and exit without creating database entries for available interfaces if no legacy data was imported. If the database already exists then access to it is only verified. The daemon will not stay running when this option is used. This option cannot be used in combination with **-d, --daemon**, **-n, --nodaemon** or **--startempty**.

--noadd
When used in combination with **-d, --daemon** or **-n, --nodaemon**, disable the automatic creation of new database entries for all currently available interfaces when the daemon is started with no existing database or with a database containing zero interfaces. The daemon will still create an empty database if one doesn't already exist. Pseudo interfaces lo, lo0 and sit0 are always excluded from getting added regardless of this option.

When used in combination with **--initdb**, create only an empty database if one doesn't already exist without importing data from possible legacy databases and exit.

-n, --nodaemon
Stay in foreground attached to the current terminal and start the update process.

--noremove
Disable automatic removal of interfaces from database that aren't currently visible and haven't seen any traffic.

-p, --pidfile *file*
Write the process id to *file* and use it for locking so that another instance of the daemon cannot be started if the same *file* is specified. This option has no effect if used in combination with **-n, --nodaemon**.

--startempty
Start even when no interfaces were discovered and the database is empty. Results in the daemon staying running and waiting for interfaces to be added to the database or found if **--alwaysadd** option has also been used. This option cannot be used in combination with **--initdb**.

-s, --sync
Synchronize internal counters in the database with interface counters for all available interfaces before starting traffic monitoring. Use this option if the traffic between the previous shutdown and the current startup of the daemon needs to be ignored. This option isn't required in normal use because the daemon will automatically synchronize the internal counters after a system reboot, if enough time has passed since the daemon was previously running or if the internal counters are clearly out of sync.

-t, --timestamp

Add a timestamp to the beginning of every print from the daemon when the process is running in the foreground attached to a terminal after having been started with the **-n, --nodaemon** option.

-u, --user *user*

Set daemon process user to *user* during startup. *user* can be either the login of the user or a numerical user id. This option can only be used when the process is started as root.

-v, --version

Show current version of the daemon executable.

-?, --help

Show a command option summary.

CONFIGURATION

The behaviour of the daemon is configured mainly using the configuration keywords **UpdateInterval, PollInterval** and **SaveInterval** in the configuration file.

UpdateInterval defines in seconds how often the interface data is fetched and updated. This is similar to the run interval for alternative cron based updating. However, the difference is that the data doesn't directly get written to disk during updates.

PollInterval defines in seconds how often the list of available interfaces is checked for possible changes. The minimum value is 2 seconds and the maximum 60 seconds. **PollInterval** also defines the resolution for other intervals.

SaveInterval defines in minutes how often cached interface data is written to disk. A write can only occur during the updating of interface data. Therefore, the value should be a multiple of **UpdateInterval** with a maximum value of 60 minutes.

The default values of **UpdateInterval** 30, **SaveInterval** 5 and **PollInterval** 5 are usually suitable for most systems and provide a similar behaviour as cron based updating does but with a better resolution for interface changes and fast interfaces.

For embedded and/or low power systems more tuned configurations are possible. In such cases if the interfaces are mostly static the **PollInterval** can be increased to around 10-30 seconds and **UpdateInterval** set to 60 seconds. Higher values up to 300 seconds are possible if the interface speed is 10 Mbit or less. **SaveInterval** can be increased for example to 15, 30 or even 60 minutes depending on how often the data needs to be viewed.

SIGNALS

The daemon is listening to signals **SIGHUP, SIGINT** and **SIGTERM.** Sending the **SIGHUP** signal to the daemon will cause cached data to be written to disk, a rescan of the database directory and a reload of settings from the configuration file. However, the pid file location will not be changed even if it's configuration setting has been modified.

SIGTERM and **SIGINT** signals will cause the daemon to write all cached data to disk and then exit.

FILES

/var/lib/vnstat/

Default database directory.

/etc/vnstat.conf

Config file that will be used unless *$HOME/.vnstatrc* exists. See the configuration chapter and **vnstat.conf**(5) for more information.

/var/log/vnstat/vnstat.log

Log file that will be used if logging to file is enable and no other file is specified in the config file.

/var/run/vnstat/vnstat.pid

File used for storing the process id when running as a background process and if no other file is specified in the configuration file or using the command line parameter.

RESTRICTIONS

Updates need to be executed at least as often as it is possible for the interface to generate enough traffic to overflow the kernel interface traffic counter. Otherwise, it is possible that some traffic won't be seen. With 32-bit interface traffic counters, the maximum time between two updates depends on how fast the interface can transfer 4 GiB. Note that there is no guarantee that a 64-bit kernel has 64-bit interface traffic counters for all interfaces. Calculated theoretical times are:

10 Mbit:	54 minutes
100 Mbit:	5 minutes
1000 Mbit:	30 seconds

Virtual and aliased interfaces cannot be monitored because the kernel doesn't provide traffic information for that type of interfaces. Such interfaces are usually named eth0:0, eth0:1, eth0:2 etc. where eth0 is the actual interface being aliased.

AUTHOR

Teemu Toivola <tst at iki dot fi>

SEE ALSO

vnstat(1), **vnstati**(1), **vnstat.conf**(5), **signal**(7)