NAME

vnStat daemon – the alternative for cron based updating

SYNOPSIS

vnstatd [**–Ddnpsv?**] [**––config** *file*] [**––daemon**] [**––debug**] [**––help**] [**––noadd**] [**––nodaemon**] [**––pidfile** *file*] [**––sync**] [**––version**]

DESCRIPTION

The purpose of **vnstatd** is to provide a more flexible way for updating **vnstat**(1) databases than what using cron for updating can provide. The daemon makes possible updating databases more often but at the same time requires less disk access since data can be cached and written only later to disk at a user configurable interval. It is also able to track how interfaces come and go without the need of additional scripts that are required with cron based updates.

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 to a user selectable file or using syslog.

Once started, the daemon will check if there are any databases available in the database directory that has been specified in the configuration file. New databases will be created for all available interfaces excluding pseudo interfaces lo, lo0 and sit0 if no databases are found during startup.

OPTIONS

--config *file*
: Use *file* as config file instead of using normal config file search function.

-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.

--noadd
: Disable automatic creation of new databases for all available interfaces if the daemon is started with zero database found. Pseudo interfaces lo, lo0 and sit0 are excluded from getting added.

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

-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.

-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 enought time has passed since the daemon was previously running or if the internal counters are clearly out of sync.

-v, --version
Show current version.

-?, --help
Show a command 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 updated. This is similar to the run interval for alternative cron based updating. However, the difference is that the data doesn't 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** 20, **SaveInterval** 5 and **PollInterval** 2 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 rised 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 rised 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 will not be updated even if it's configuration setting has been changed.

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. Files are named according to the monitored interfaces.

/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.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.pid
: File used for storing the process id if no other file is specified in the configuration file or using the command line parameter.

RESTRICTIONS

Updates needs to be executed at least as often as it is possible for the interface to generate enough traffic to wrap the kernel interface traffic counter. Otherwise it is possible that some traffic won't be seen. This isn't an issue for 64 bit kernels but at least one update every hour is always required in order to provide proper input. With 32 bit kernels the maximum time between two updates depends on how fast the interface can transfer 4 GiB. Calculated theoretical times are:

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

However, for 1000 Mbit interfaces updating once every minute is usually a working solution.

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)