NAME

smartd − SMART Disk Monitoring Daemon

SYNOPSIS

smartd [options]

DESCRIPTION

[This man page is generated for the Windows version of smartmontools. It does not contain info specific to other platforms.]

smartd is a daemon that monitors the Self-Monitoring, Analysis and Reporting Technology (SMART) system built into most ATA/SATA and SCSI/SAS hard drives and solid-state drives. The purpose of SMART is to monitor the reliability of the hard drive and predict drive failures, and to carry out different types of drive self-tests. This version of **smartd** is compatible with ACS-3, ACS-2, ATA8-ACS, ATA/ATAPI-7 and earlier standards (see **REFERENCES** below).

smartd will attempt to enable SMART monitoring on ATA devices (equivalent to **smartctl −s on**) and polls these and SCSI devices every 30 minutes (configurable), logging SMART errors and changes of SMART Attributes via the SYSLOG interface. The default location for these SYSLOG notifications and warnings is system-dependent (typically **/var/log/messages** or **/var/log/syslog**). To change this default location, please see the '−l' command-line option described below.

In addition to logging to a file, **smartd** can also be configured to send email warnings if problems are detected. Depending upon the type of problem, you may want to run self-tests on the disk, back up the disk, replace the disk, or use a manufacturer's utility to force reallocation of bad or unreadable disk sectors. If disk problems are detected, please see the **smartctl** manual page and the **smartmontools** web page/FAQ for further guidance.

If you send a **USR1** signal to **smartd** it will immediately check the status of the disks, and then return to polling the disks every 30 minutes. See the '−i' option below for additional details.

smartd can be configured at start-up using the configuration file **/usr/local/etc/smartd.conf** (Windows: **EXEDIR/smartd.conf**). If the configuration file is subsequently modified,**smartd** can be told to re-read the configuration file by sending it a **HUP** signal, for example with the command:
killall −HUP smartd.
(Windows: See NOTES below.)

On startup, if **smartd** finds a syntax error in the configuration file, it will print an error message and then exit. However if **smartd** is already running, then is told with a **HUP** signal to re-read the configuration file, and then find a syntax error in this file, it will print an error message and then continue, ignoring the contents of the (faulty) configuration file, as if the **HUP** signal had never been received.

When **smartd** is running in debug mode, the **INT** signal (normally generated from a shell with CONTROL-C) is treated in the same way as a **HUP** signal: it makes **smartd** reload its configuration file. To exit **smartd** use CONTROL-\. (Windows: CONTROL-Break).

On startup, in the absence of the configuration file **/usr/local/etc/smartd.conf**, the **smartd** daemon first scans for all devices that support SMART. The scanning is done as follows:

WINDOWS:

Examine all entries **"/dev/sd[a−z]"**, **"/dev/sd[a−c][a−z]"** and **"/dev/sdd[a−x]"** ("\\.\PhysicalDrive[0−127]") for IDE/(S)ATA and SCSI disk devices.

If a 3ware 9000 controller is installed, examine all entries **"/dev/sdX,N"** for the first logical drive ('unit' **"/dev/sdX"**) and all physical disks ('ports' **",N"**) detected behind this controller. Same for a second controller if present.

If directive '−d csmi' or no '−d' directive is specified, examine all entries **"/dev/csmi[0−9],N"** for drives behind an Intel ICHxR controller with RST driver.

Disks behind Areca RAID controllers are not included.

If directive '−d nvme' or no '−d' directive is specified, examine all entries **"/dev/sd[...]"** (see

above) and all entries **"/dev/nvme[0−9]"** for NVMe devices.

smartd then monitors for *all* possible SMART errors (corresponding to the '−a' Directive in the configuration file; see the **smartd.conf**(5) man page).

OPTIONS

−A PREFIX, −−attributelog=PREFIX

Writes drive attribute information to files
'PREFIX"MODEL−SERIAL[−NSID].PRT.csv'.
PREFIX specifies the destination directory if it includes a trailing slash. Otherwise its last path component specifies the prefix of the filename. The path must be absolute, except if debug mode is enabled. MODEL and SERIAL are build from drive identify information, invalid characters are replaced by underline. NSID is the NVMe namespace id if an individual namespace is addressed. PRT is one of 'ata', 'scsi' or 'nvme'.

At each check cycle, attributes are logged as a line of tab separated tuples which consists of semicolon separated names or values. Each line is led by a date string of the form "yyyy-mm-dd HH:MM:SS" (in local time). The remaining part is protocol-specific:
[ATA] Writes ATA SMART attributes as "id;normalized-value;raw-value;".
[SCSI] Writes SCSI error counters and temperature as "name;value;".
[NVMe: NEW EXPERIMENTAL SMARTD 7.5 FEATURE] Writes NVMe SMART/Health information as "name;value;".

−B [+]FILE, −−drivedb=[+]FILE

[ATA only] Read the drive database from FILE. The new database replaces the built in database by default. If '+' is specified, then the new entries prepend the built in entries. Please see the **smartctl**(8) man page for further details.

−c FILE, −−configfile=FILE

Read **smartd** configuration Directives from FILE, instead of from the default location **/usr/local/etc/smartd.conf** (Windows: **EXEDIR/smartd.conf**). If FILE does**not** e xist, then **smartd** will print an error message and exit with nonzero status. Thus, '−c /usr/local/etc/smartd.conf' can be used to verify the existence of the default configuration file.

By using '−' for FILE, the configuration is read from standard input. This is useful for commands like:
echo /dev/sdb −m user@home −M test | smartd −c − −q onecheck
to perform quick and simple checks without a configuration file.

−d, −−debug

Runs **smartd** in "debug" mode. In this mode, it displays status information to STDOUT rather than logging it to SYSLOG and does not **fork**(2) into the background and detach from the controlling terminal. In this mode, **smartd** also prints more verbose information about what it is doing than when operating in "daemon" mode. In this mode, the **INT** signal (normally generated from a terminal with CONTROL-C) makes **smartd** reload its configuration file. Please use CONTROL-\ to exit (Windows: CONTROL-Break).

[Windows only] The "debug" mode can be toggled by the command **smartd sigusr2**. A new console for debug output is opened when debug mode is enabled.

−D, −−showdirectives

Prints a list (to STDOUT) of all the possible Directives which may appear in the configuration file /usr/local/etc/smartd.conf, and then exits. These Directives are described in the **smartd.conf**(5) man page. They may appear in the configuration file following the device name.

−h, −−help, −−usage

Prints usage message to STDOUT and exits.

−i N, −−interval=N

Sets the interval between disk checks to *N* seconds, where *N* is a decimal integer. The minimum allowed value is ten and the maximum is the largest positive integer that can be represented on

your system (often 2^31−1). The default is 1800 seconds. The interval could be overridden with the '−c i=N' directive, see **smartd.conf**(5) man page.

Note that the superuser can make **smartd** check the status of the disks at any time by sending it the **SIGUSR1** signal, for example with the command:
kill −SIGUSR1 <pid>
where **<pid>** is the process id number of **smartd**. One may also use:
killall −USR1 smartd
for the same purpose.
(Windows: See NOTES below.)

−l FACILITY, −−logfacility=FACILITY

Uses syslog facility FACILITY to log the messages from **smartd**. Here FACILITY is one of *local0*, *local1*, ..., *local7*, or *daemon* [default]. If this command-line option is not used, then by default messages from **smartd** are logged to the facility *daemon*.

If you would like to have **smartd** messages logged somewhere other than the default location, include (for example) '−l local3' in its start up argument list. Tell the syslog daemon to log all messages from facility **local3** to (for example) '/var/log/smartd.log'.

For more detailed information, please refer to the man pages for the local syslog daemon, typically **syslogd**(8), **syslog-ng**(8) or **rsyslogd**(8).

Windows: Some **syslog** functionality is implemented internally in **smartd** as follows: If no '−l' option (or '−l daemon') is specified, messages are written to Windows event log or to file **./smartd.log** if event log is not available (access denied). By specifying other values of FACILITY, log output is redirected as follows: '−l local0' to file **./smartd.log**, '−l local1' to standard output (redirect with '>' to any file), '−l local2' to standard error, '−l local[3−7]': to file **./smartd[1−5].log**.

−n, −−no−fork

Do not fork into background; this is useful when executed from modern init methods like initng, minit, supervise or systemd.

On Windows, this option is not available, use '−−service' instead.

−p NAME, −−pidfile=NAME

Writes pidfile *NAME* containing the **smartd** Process ID number (PID). To avoid symlink attacks make sure the directory to which pidfile is written is only writable for root. Without this option, or if the −−debug option is given, no PID file is written on startup. If **smartd** is killed with a maskable signal then the pidfile is removed.

−q WHEN, −−quit=WHEN

Specifies when, if ever, **smartd** should exit. The valid arguments are to this option are:

nodev − Exit if there are no devices to monitor, or if any errors are found at startup in the configuration file. This is the default.

errors − Exit if there are no devices to monitor, or if any errors are found in the configuration file /usr/local/etc/smartd.conf at startup or whenever it is reloaded.

nodevstartup − Exit if there are no devices to monitor at startup. But continue to run if no devices are found whenever the configuration file is reloaded.

never − Only exit if a fatal error occurs (no remaining system memory, invalid command line arguments). In this mode, even if there are no devices to monitor, or if the configuration file **/usr/local/etc/smartd.conf** has errors, **smartd** will continue to run, waiting to load a configuration file listing valid devices.

nodev0 − Same as 'nodev', except that the exit status is 0 if there are no devices to monitor.

nodev0startup − Same as 'nodevstartup', except that the exit status is 0 if there are no devices to monitor.

errors,nodev0 − Same as 'errors', except that the exit status is 0 if there are no devices to monitor.

onecheck − Start **smartd** in debug mode, then register devices, then check device's SMART status once, and then exit with zero exit status if all of these steps worked correctly.

This last option is intended for 'distribution-writers' who want to create automated scripts to determine whether or not to automatically start up **smartd** after installing smartmontools. After starting **smartd** with this command-line option, the distribution's install scripts should wait a reasonable length of time (say ten seconds). If **smartd** has not exited with zero status by that time, the script should send **smartd** a SIGTERM or SIGKILL and assume that **smartd** will not operate correctly on the host. Conversely, if **smartd** exits with zero status, then it is safe to run **smartd** in normal daemon mode. If **smartd** is unable to monitor any devices or encounters other problems then it will return with non-zero exit status.

showtests − Start **smartd** in debug mode, then register devices, then write a list of future scheduled self tests to stdout, and then exit with zero exit status if all of these steps worked correctly. Device's SMART status is not checked.

This option is intended to test whether the '−s REGEX' directives in smartd.conf will have the desired effect. The output lists the next test schedules, limited to 5 tests per type and device. This is followed by a summary of all tests of each device within the next 90 days.

−r TYPE, −−report=TYPE

Intended primarily to help **smartmontools** developers understand the behavior of **smartmontools** on non-conforming or poorly-conforming hardware. This option reports details of**smartd** transactions with the device. The option can be used multiple times. When used just once, it shows a record of the ioctl() transactions with the device. When used more than once, the detail of these ioctl() transactions are reported in greater detail. The valid arguments to this option are:

ioctl − report all ioctl() transactions.

ataioctl − report only ioctl() transactions with ATA devices.

scsiioctl − report only ioctl() transactions with SCSI devices.

nvmeioctl − report only ioctl() transactions with NVMe devices.

Any argument may include a positive integer to specify the level of detail that should be reported. The argument should be followed by a comma then the integer with no spaces. For example, *ataioctl,2* The default level is 1, so '−r ataioctl,1' and '−r ataioctl' are equivalent.

−s PREFIX, −−savestates=PREFIX

Reads/writes **smartd** state information from/to files
'PREFIX"MODEL−SERIAL[−NSID].PRT.state'.
For PREFIX, MODEL, SERIAL, NSID and PRT, see the '-A' option above.

The state information files preserve various information across smartd restarts, for example: SMART attributes, drive min and max temperatures ('−W' directive), info about last sent warning email ('−m' directive), the time of next check of the self-test REGEXP ('−s' directive), and info about the last reported error log entries ('−l [x]error' directives).

The state information files are read on smartd startup. The files are always (re)written after reading the configuration file, before rereading the configuration file (SIGHUP), before smartd shutdown, and after a check forced by SIGUSR1. After a normal check cycle, a file is only rewritten if an important change (which usually results in a SYSLOG output) occurred.

−w PATH, −−warnexec=PATH

Run the executable PATH instead of the default script when smartd needs to send warning messages. PATH must point to an executable binary file or script. The default script is **EXEDIR/smartd_warning.cmd**.

−u MODE, −−warn−as−user=MODE
[Windows only] Run the warning script with a modified access token. The valid arguments to this option are:

restricted − Run the warning script with a restricted access token. The local 'Administrator' group and most privileges (all except 'SeChangeNotifyPrivilege') are removed. This is not effective if the current user is the local 'SYSTEM' or 'Administrator' account. If this is the case, **smartd** logs an error message during startup and exits.

unchanged − Run the warning script without changing the access token. This is the default.

−−service
[Windows only] Enables **smartd** to run as a Windows service. The option must be specified in the service command line as the first argument. It should not be used from console. See NOTES below for details.

−V, −−version, −−license, −−copyright
Prints version, copyright, license, home page and SVN revision information for your copy of **smartd** to STDOUT and then exits.

EXAMPLES

smartd
Runs the daemon in forked mode. This is the normal way to run **smartd**. Entries are logged to SYSLOG.

smartd −d −i 30
Run in foreground (debug) mode, checking the disk status every 30 seconds.

smartd −q onecheck
Registers devices, and checks the status of the devices exactly once. The exit status (the shell **$?** variable) will be zero if all went well, and nonzero if no devices were detected or some other problem was encountered.

CONFIGURATION

The syntax of the **smartd.conf**(5) file is discussed separately.

NOTES

smartd will make log entries at loglevel **LOG_INFO** if the Normalized SMART Attribute values have changed, as reported using the **'−t', '−p',** or **'−u'** Directives. For example:
'Device: /dev/sda, SMART Attribute: 194 Temperature_Celsius changed from 94 to 93'
Note that in this message, the value given is the 'Normalized' not the 'Raw' Attribute value (the disk temperature in this case is about 22 Celsius). The **'−R'** and **'−r'** Directives modify this behavior, so that the information is printed with the Raw values as well, for example:
'Device: /dev/sda, SMART Attribute: 194 Temperature_Celsius changed from 94 [Raw 22] to 93 [Raw 23]'
Here the Raw values are the actual disk temperatures in Celsius. The way in which the Raw values are printed, and the names under which the Attributes are reported, is governed by the various **'−v Num,Description'** Directives described previously.

Please see the **smartctl** manual page for further explanation of the differences between Normalized and Raw Attribute values.

smartd will make log entries at loglevel **LOG_CRIT** if a SMART Attribute has failed, for example:
'Device: /dev/sdc, Failed SMART Attribute: 5 Reallocated_Sector_Ct'
This loglevel is used for reporting enabled by the **'−H', −f', '−l selftest',** and **'−l error'** Directives. Entries reporting failure of SMART Prefailure Attributes should not be ignored: they mean that the disk is failing. Use the **smartctl** utility to investigate.

On Windows, the log messages are written to the event log or to a file. See documentation of the '−l FACILITY' option above for details.

On Windows, the following built-in commands can be used to control **smartd**, if running as a daemon:

'**smartd status**' − check status
'**smartd stop**' − stop smartd
'**smartd reload**' − reread config file
'**smartd restart**' − restart smartd
'**smartd sigusr1**' − check disks now
'**smartd sigusr2**' − toggle debug mode

The Windows Version of **smartd** has buildin support for services:

'**smartd install [options]**' installs a service named "smartd" (display name "SmartD Service") using the command line '/INSTALLPATH/smartd.exe −−service [options]'. This also installs smartd.exe as a event message file for the Windows event viewer.

This does not work if the option '--warn-as-user=restricted' is specified because the local 'SYSTEM' account cannot be restricted. The service must then be manually reconfigured to run as a another user which is a member of the local 'Administrator' group.

'**smartd remove**' can later be used to remove the service and event message entries from the registry.

Upon startup, the smartd service changes the working directory to its own installation path. If smartd.conf and blat.exe are stored in this directory, no '−c' option and '−M exec' directive is needed.

The debug mode ('−d', '−q onecheck') does not work if smartd is running as service.

The service can be controlled as usual with Windows commands 'net' or 'sc' ('**net start smartd**', '**net stop smartd**').

Pausing the service ('**net pause smartd**') sets the interval between disk checks ('−i N') to infinite.

Continuing the paused service ('**net continue smartd**') resets the interval and rereads the configuration file immediately (like **SIGHUP**). The 'PARAMCHANGE' service control command ('**sc control smartd paramchange**') has the same effect regardless of paused state.

Continuing a still running service ('**net continue smartd**' without preceding '**net pause smartd**') does not reread configuration but checks disks immediately (like **SIGUSR1**).

LOG TIMESTAMP TIMEZONE

When **smartd** makes log entries, these are time-stamped. The time stamps are in the computer's local time zone, which is generally set using either the environment variable '**TZ**' or using a time-zone file such as **/etc/localtime**. You may wish to change the timezone while **smartd** is running (for example, if you carry a laptop to a new time-zone and don't reboot it). Due to a bug in the **tzset**(3) function of many unix standard C libraries, the time-zone stamps of **smartd** might not change. For some systems, **smartd** will work around this problem *if* the time-zone is set using **/etc/localtime**. The work-around *fails* if the time-zone is set using the '**TZ**' variable (or a file that it points to).

EXIT STATUS

The exit status (return value) of **smartd** can have the following values:

0: Daemon startup successful, or **smartd** was killed by a SIGTERM (or in debug mode, a SIGQUIT).

1: Commandline did not parse.

2: There was a syntax error in the config file.

3: Forking the daemon failed.

4: Couldn't create PID file.

5: Config file does not exist (only returned in conjunction with the '−c' option).

6: Config file exists, but cannot be read.

8: **smartd** ran out of memory during startup.

10: An inconsistency was found in **smartd**'s internal data structures. This should never happen. It must be due to either a coding or compiler bug. *Please* report such failures to smartmontools developers, see REPORTING BUGS below.

16: A device explicitly listed in **/usr/local/etc/smartd.conf** can't be monitored.

17: **smartd** didn't find any devices to monitor. This could be changed to**0** (success) with one of the '-q *nodev0*' options, see above.

254: When in daemon mode, **smartd** received a SIGINT or SIGQUIT. (Note that in debug mode, SIGINT has the same effect as SIGHUP, and makes **smartd** reload its configuration file. SIGQUIT has the same effect as SIGTERM and causes **smartd** to exit with zero exit status.

132 and above
smartd was killed by a signal that is not explicitly listed above. The exit status is then 128 plus the signal number. For example if **smartd** is killed by SIGKILL (signal 9) then the exit status is 137.

AUTHORS

Bruce Allen (project initiator),
Christian Franke (project manager, Windows port and all sort of things),
Douglas Gilbert (SCSI subsystem),
Volker Kuhlmann (moderator of support and database mailing list),
Gabriele Pohl (wiki & development team support),
Alex Samorukov (FreeBSD port and more, new Trac wiki).

Many other individuals have made contributions and corrections, see AUTHORS, ChangeLog and repository files.

The first smartmontools code was derived from the smartsuite package, written by Michael Cornwell and Andre Hedrick.

REPORTING BUGS

To submit a bug report, create a ticket in smartmontools wiki:
<**https://www.smartmontools.org/**>.
Alternatively send the info to the smartmontools support mailing list:
<**https://listi.jpberlin.de/mailman/listinfo/smartmontools-support>.**

SEE ALSO

smartd.conf(5), **smartctl**(8).
update-smart-drivedb(8).

REFERENCES

Please see the following web site for more info: <**https://www.smartmontools.org/**>

An introductory article about smartmontools is *Monitoring Hard Disks with SMART*, by Bruce Allen, Linux Journal, January 2004, pages 74–77. See <**https://www.linuxjournal.com/article/6983**>.

If you would like to understand better how SMART works, and what it does, a good place to start is with Sections 4.8 and 6.54 of the first volume of the 'AT Attachment with Packet Interface-7' (ATA/ATAPI-7) specification Revision 4b. This documents the SMART functionality which the **smartmontools** utilities provide access to.

The functioning of SMART was originally defined by the SFF-8035i revision 2 and the SFF-8055i revision 1.4 specifications. These are publications of the Small Form Factors (SFF) Committee.

Links to these and other documents may be found on the Links page of the **smartmontools** Wiki at <**https://www.smartmontools.org/wiki/Links**>.

PACKAGE VERSION

smartmontools-7.5 2025-04-30 r5714
Id