NAME

watchdogd - watchdog daemon

SYNOPSIS

watchdogd [**-dnSw**] [**--debug**] [**--softtimeout**] [**--softtimeout-action** *action*] [**--pretimeout** *timeout*] [**--pretimeout-action** *action*] [**-e** *cmd*] [**-I** *file*] [**-s** *sleep*] [**-t** *timeout*] [**-T** *script_timeout*] [**-x** *exit_timeout*]

DESCRIPTION

The **watchdogd** utility interfaces with the kernel's watchdog facility to ensure that the system is in a working state. If **watchdogd** is unable to interface with the kernel over a specific timeout, the kernel will take actions to assist in debugging or restarting the computer.

If **-e** *cmd* is specified, **watchdogd** will attempt to execute this command with system(3), and only if the command returns with a zero exit code will the watchdog be reset. If **-e** *cmd* is not specified, the daemon will perform a trivial file system check instead.

The **-n** argument 'dry-run' will cause watchdog not to arm the system watchdog and instead only run the watchdog function and report on failures. This is useful for developing new watchdogd scripts as the system will not reboot if there are problems with the script.

The **-s** *sleep* argument can be used to control the sleep period between each execution of the check and defaults to 10 seconds.

The **-t** *timeout* specifies the desired timeout period in seconds. The default timeout is 128 seconds.

One possible circumstance which will cause a watchdog timeout is an interrupt storm. If this occurs, **watchdogd** will no longer execute and thus the kernel's watchdog routines will take action after a configurable timeout.

The **-T** *script_timeout* specifies the threshold (in seconds) at which the watchdogd will complain that its script has run for too long. If unset *script_timeout* defaults to the value specified by the **-s** *sleep* option.

The **-x** *exit_timeout* argument is the timeout period (in seconds) to leave in effect when the program exits. Using **-x** with a non-zero value protects against lockup during a reboot by triggering a hardware reset if the software reboot doesn't complete before the given timeout expires.

Upon receiving the SIGTERM or SIGINT signals, **watchdogd** will terminate, after first instructing the kernel to either disable the timeout or reset it to the value given by **-x** *exit_timeout*.

The **watchdogd** utility recognizes the following runtime options:

-I *file*
: Write the process ID of the **watchdogd** utility in the specified file.

-d --debug
: Do not fork. When this option is specified, **watchdogd** will not fork into the background at startup.

-S
: Do not send a message to the system logger when the watchdog command takes longer than expected to execute. The default behaviour is to log a warning via the system logger with the LOG_DAEMON facility, and to output a warning to standard error.

-w
: Complain when the watchdog script takes too long. This flag will cause watchdogd to complain when the amount of time to execute the watchdog script exceeds the threshold of 'sleep' option.

--pretimeout *timeout*
: Set a "pretimeout" watchdog. At "timeout" seconds before the watchdog will fire attempt an action. The action is set by the --pretimeout-action flag. The default is just to log a message (WD_SOFT_LOG) via log(9).

--pretimeout-action *action*
: Set the timeout action for the pretimeout. See the section *Timeout Actions*.

--softtimeout
: Instead of arming the various hardware watchdogs, only use a basic software watchdog. The default action is just to log(9) a message (WD_SOFT_LOG).

--softtimeout-action *action*
: Set the timeout action for the softtimeout. See the section *Timeout Actions*.

Timeout Actions

The following timeout actions are available via the **--pretimeout-action** and **--softtimeout-action** flags:

panic Call panic(9) when the timeout is reached.

ddb Enter the kernel debugger via kdb_enter(9) when the timeout is reached.

log Log a message using log(9) when the timeout is reached.

printf call the kernel printf(9) to display a message to the console and dmesg(8) buffer.

Actions can be combined in a comma separated list as so: *log,printf* which would both printf(9) and log(9) which will send messages both to dmesg(8) and the kernel log(4) device for syslogd(8).

FILES

/var/run/watchdogd.pid

EXAMPLES

Debugging watchdogd and/or your watchdog script.

This is a useful recipe for debugging **watchdogd** and your watchdog script.

(Note that ^C works oddly because **watchdogd** calls system(3) so the first ^C will terminate the "sleep" command.)

Explanation of options used:

1. Set Debug on (--debug)
2. Set the watchdog to trip at 30 seconds. (-t 30)
3. Use of a softtimeout:
    1. Use a softtimeout (do not arm the hardware watchdog). (--softtimeout)
    2. Set the softtimeout action to do both kernel printf(9) and log(9) when it trips. (--softtimeout-action log,printf)
4. Use of a pre-timeout:
    1. Set a pre-timeout of 15 seconds (this will later trigger a panic/dump). (--pretimeout 15)
    2. Set the action to also kernel printf(9) and log(9) when it trips. (--pretimeout-action log,printf)
5. Use of a script:
    1. Run "sleep 60" as a shell command that acts as the watchdog (-e 'sleep 60')
    2. Warn us when the script takes longer than 1 second to run (-w)

```
watchdogd --debug -t 30 \
  --softtimeout --softtimeout-action log,printf \
  --pretimeout 15 --pretimeout-action log,printf \
  -e 'sleep 60' -w
```

Production use of example

1. Set hard timeout to 120 seconds (-t 120)
2. Set a panic to happen at 60 seconds (to trigger a crash(8) for dump analysis):
    1. Use of pre-timeout (--pretimeout 60)
    2. Specify pre-timeout action (--pretimeout-action log,printf,panic)
3. Use of a script:
    1. Run your script (-e '/path/to/your/script 60')
    2. Log if your script takes a longer than 15 seconds to run time. (-w -T 15)

```
watchdogd  -t 120 \
  --pretimeout 60 --pretimeout-action log,printf,panic \
  -e '/path/to/your/script 60' -w -T 15
```

SEE ALSO

watchdog(4), watchdog(8), watchdog(9)

HISTORY

The **watchdogd** utility appeared in FreeBSD 5.1.

AUTHORS

The **watchdogd** utility and manual page were written by Sean Kelly <*smkelly@FreeBSD.org*> and Poul-Henning Kamp <*phk@FreeBSD.org*>.

Some contributions made by Jeff Roberson <*jeff@FreeBSD.org*>.

The pretimeout and softtimeout action system was added by Alfred Perlstein <*alfred@freebsd.org*>.