NAME

Sys::Syslog - Perl interface to the UNIX syslog(3) calls

VERSION

Version 0.13

SYNOPSIS

```
    use Sys::Syslog;                        # all except setlogsock(),
or:
    use Sys::Syslog qw(:DEFAULT setlogsock);  # default set, plus
setlogsock()
    use Sys::Syslog qw(:standard :macros);    # standard functions, plus
macros

    setlogsock $sock_type;
    openlog $ident, $logopt, $facility;       # don't forget this
    syslog $priority, $format, @args;
    $oldmask = setlogmask $mask_priority;
    closelog;
```

DESCRIPTION

`Sys::Syslog` is an interface to the UNIX `syslog(3)` program. Call `syslog()` with a string priority and a list of `printf()` args just like `syslog(3)`.

EXPORTS

`Sys::Syslog` exports the following `Exporter` tags:

- `:standard` exports the standard `syslog(3)` functions:

  ```
  openlog closelog setlogmask syslog
  ```

- `:extended` exports the Perl specific functions for `syslog(3)`:

  ```
  setlogsock
  ```

- `:macros` exports the symbols corresponding to most of your `syslog(3)` macros. See *CONSTANTS* for the supported constants and their meaning.

By default, `Sys::Syslog` exports the symbols from the `:standard` tag.

FUNCTIONS

openlog($ident, $logopt, $facility)

Opens the syslog. `$ident` is prepended to every message. `$logopt` contains zero or more of the words `pid`, `ndelay`, `nowait`. The `cons` option is ignored, since the failover mechanism will drop down to the console automatically if all other media fail. `$facility` specifies the part of the system to report about, for example `LOG_USER` or `LOG_LOCAL0`: see your `syslog(3)` documentation for the facilities available in your system. Facility can be given as a string or a numeric macro.

This function will croak if it can't connect to the syslog daemon.

Note that `openlog()` now takes three arguments, just like `openlog(3)`.

You should use openlog() before calling syslog().

Options

- `ndelay` - Open the connection immediately (normally, the connection is opened when the first message is logged).

- `nowait` - Don't wait for child processes that may have been created while logging the message. (The GNU C library does not create a child process, so this option has no effect on Linux.)
- `pid` - Include PID with each message.

Examples

Open the syslog with options `ndelay` and `pid`, and with facility `LOCAL0`:

```
openlog($name, "ndelay,pid", "local0");
```

Same thing, but this time using the macro corresponding to `LOCAL0`:

```
openlog($name, "ndelay,pid", LOG_LOCAL0);
```

syslog($priority, $message)

syslog($priority, $format, @args)

If `$priority` permits, logs `$message` or `sprintf($format, @args)` with the addition that `%m` in $message or `$format` is replaced with `"$!"` (the latest error message).

`$priority` can specify a level, or a level and a facility. Levels and facilities can be given as strings or as macros.

If you didn't use `openlog()` before using `syslog()`, `syslog()` will try to guess the `$ident` by extracting the shortest prefix of `$format` that ends in a `":"`.

Examples

```
    syslog("info", $message);           # informational level
    syslog(LOG_INFO, $message);         # informational level

    syslog("info|local0", $message);        # information level,
Local0 facility
    syslog(LOG_INFO|LOG_LOCAL0, $message);  # information level,
Local0 facility
```

Note

`Sys::Syslog` version v0.07 and older passed the `$message` as the formatting string to `sprintf()` even when no formatting arguments were provided. If the code calling `syslog()` might execute with older versions of this module, make sure to call the function as `syslog($priority, "%s", $message)` instead of `syslog($priority, $message)`. This protects against hostile formatting sequences that might show up if $message contains tainted data.

setlogmask($mask_priority)

Sets the log mask for the current process to `$mask_priority` and returns the old mask. If the mask argument is 0, the current log mask is not modified. See *Levels* for the list of available levels.

Examples

Only log errors:

```
setlogmask(LOG_ERR);
```

Log critical messages, errors and warnings:

```
setlogmask(LOG_CRIT|LOG_ERR|LOG_WARNING);
```

setlogsock($sock_type)

setlogsock($sock_type, $stream_location) (added in 5.004_02)

Sets the socket type to be used for the next call to `openlog()` or `syslog()` and returns true on success, `undef` on failure.

A value of `"unix"` will connect to the UNIX domain socket (in some systems a character special device) returned by the `_PATH_LOG` macro (if your system defines it), or */dev/log* or */dev/conslog*, whatever is writable. A value of 'stream' will connect to the stream indicated by the pathname provided as the optional second parameter. (For example Solaris and IRIX require `"stream"` instead of `"unix"`.) A value of `"inet"` will connect to an INET socket (either `tcp` or `udp`, tried in that order) returned by `getservbyname()`. `"tcp"` and `"udp"` can also be given as values. The value `"console"` will send messages directly to the console, as for the `"cons"` option in the logopts in `openlog()`.

A reference to an array can also be passed as the first parameter. When this calling method is used, the array should contain a list of sock_types which are attempted in order.

The default is to try `tcp`, `udp`, `unix`, `stream`, `console`.

Giving an invalid value for `$sock_type` will croak.

closelog()

Closes the log file and return true on success.

EXAMPLES

```
openlog($program, 'cons,pid', 'user');
syslog('info', '%s', 'this is another test');
syslog('mail|warning', 'this is a better test: %d', time);
closelog();

syslog('debug', 'this is the last test');

setlogsock('unix');
openlog("$program $$", 'ndelay', 'user');
syslog('notice', 'fooprogram: this is really done');

setlogsock('inet');
$! = 55;
syslog('info', 'problem was %m'); # %m == $! in syslog(3)

# Log to UDP port on $remotehost instead of logging locally
setlogsock('udp');
$Sys::Syslog::host = $remotehost;
openlog($program, 'ndelay', 'user');
syslog('info', 'something happened over here');
```

CONSTANTS

Facilities

- `LOG_AUTH` - security/authorization messages
- `LOG_AUTHPRIV` - security/authorization messages (private)
- `LOG_CRON` - clock daemon (**cron** and **at**)
- `LOG_DAEMON` - system daemons without separate facility value
- `LOG_FTP` - ftp daemon
- `LOG_KERN` - kernel messages

- LOG_LOCAL0 through LOG_LOCAL7 - reserved for local use
- LOG_LPR - line printer subsystem
- LOG_MAIL - mail subsystem
- LOG_NEWS - USENET news subsystem
- LOG_SYSLOG - messages generated internally by **syslogd**
- LOG_USER (default) - generic user-level messages
- LOG_UUCP - UUCP subsystem

Levels

- LOG_EMERG - system is unusable
- LOG_ALERT - action must be taken immediately
- LOG_CRIT - critical conditions
- LOG_ERR - error conditions
- LOG_WARNING - warning conditions
- LOG_NOTICE - normal, but significant, condition
- LOG_INFO - informational message
- LOG_DEBUG - debug-level message

DIAGNOSTICS

Invalid argument passed to setlogsock

(F) You gave `setlogsock()` an invalid value for `$sock_type`.

no connection to syslog available

(F) `syslog()` failed to connect to the specified socket.

stream passed to setlogsock, but %s is not writable

(W) You asked `setlogsock()` to use a stream socket, but the given path is not writable.

stream passed to setlogsock, but could not find any device

(W) You asked `setlogsock()` to use a stream socket, but didn't provide a path, and `Sys::Syslog` was unable to find an appropriate one.

tcp passed to setlogsock, but tcp service unavailable

(W) You asked `setlogsock()` to use a TCP socket, but the service is not available on the system.

syslog: expecting argument %s

(F) You forgot to give `syslog()` the indicated argument.

syslog: invalid level/facility: %s

(F) You specified an invalid level or facility, like LOG_KERN (which is reserved to the kernel).

syslog: too many levels given: %s

(F) You specified too many levels.

syslog: too many facilities given: %s

(F) You specified too many facilities.

syslog: level must be given

(F) You forgot to specify a level.

udp passed to setlogsock, but udp service unavailable

(W) You asked `setlogsock()` to use a UDP socket, but the service is not available on the system.

unix passed to setlogsock, but path not available

(W) You asked `setlogsock()` to use a UNIX socket, but `Sys::Syslog` was unable to find an appropriate an appropriate device.

SEE ALSO

syslog(3)

Syslogging with Perl, *http://lexington.pm.org/meetings/022001.html*

AUTHOR

Tom Christiansen <*tchrist@perl.com*> and Larry Wall <*larry@wall.org*>.

UNIX domain sockets added by Sean Robinson <*robinson_s@sc.maricopa.edu*> with support from Tim Bunce <*Tim.Bunce@ig.co.uk*> and the `perl5-porters` mailing list.

Dependency on *syslog.ph* replaced with XS code by Tom Hughes <*tom@compton.nu*>.

Code for `constant()`s regenerated by Nicholas Clark <*nick@ccl4.org*>.

Failover to different communication modes by Nick Williams <*Nick.Williams@morganstanley.com*>.

Extracted from core distribution for publishing on the CPAN by Sébastien Aperghis-Tramoni <sebastien@aperghis.net>.

BUGS

Please report any bugs or feature requests to `bug-sys-syslog at rt.cpan.org`, or through the web interface at *http://rt.cpan.org/NoAuth/ReportBug.html?Queue=Sys-Syslog*. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.

SUPPORT

You can find documentation for this module with the perldoc command.

```
perldoc Sys::Syslog
```

You can also look for information at:

* AnnoCPAN: Annotated CPAN documentation

  http://annocpan.org/dist/Sys-Syslog

* CPAN Ratings

  http://cpanratings.perl.org/d/Sys-Syslog

* RT: CPAN's request tracker

  http://rt.cpan.org/NoAuth/Bugs.html?Dist=Sys-Syslog

* Search CPAN

  http://search.cpan.org/dist/Sys-Syslog

LICENSE

This program is free software; you can redistribute it and/or modify it under the same terms as Perl itself.