## NAME

**syslog**, **vsyslog**, **openlog**, **closelog**, **setlogmask** - control system log

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <syslog.h>**

*void*
**syslog**(*int priority*, *const char *message*, ...);

*void*
**openlog**(*const char *ident*, *int logopt*, *int facility*);

*void*
**closelog**(*void*);

*int*
**setlogmask**(*int maskpri*);

**#include <syslog.h>**
**#include <stdarg.h>**

*void*
**vsyslog**(*int priority*, *const char *message*, *va_list args*);

## DESCRIPTION

The **syslog**() function writes *message* to the system message logger. The message is then written to the system console, log files, logged-in users, or forwarded to other machines as appropriate. (See syslogd(8).)

The message is identical to a printf(3) format string, except that '%m' is replaced by the current error message. (As denoted by the global variable *errno*; see strerror(3).) A trailing newline is added if none is present.

The **vsyslog**() function is an alternate form in which the arguments have already been captured using the variable-length argument facilities of stdarg(3).

The message is tagged with *priority*. Priorities are encoded as a *facility* and a *level*. The facility

describes the part of the system generating the message. The level is selected from the following *ordered* (high to low) list:

LOG_EMERG A panic condition. This is normally broadcast to all users.

LOG_ALERT A condition that should be corrected immediately, such as a corrupted system database.

LOG_CRIT Critical conditions, e.g., hard device errors.

LOG_ERR Errors.

LOG_WARNING Warning messages.

LOG_NOTICE Conditions that are not error conditions, but should possibly be handled specially.

LOG_INFO Informational messages.

LOG_DEBUG Messages that contain information normally of use only when debugging a program.

The **openlog**() function provides for more specialized processing of the messages sent by **syslog**() and **vsyslog**(). The *ident* argument is a string that will be prepended to every message. It may be formatted as *ident[N]* in which case decimal number *N* replaces the process id within messages. The *logopt* argument is a bit field specifying logging options, which is formed by OR'ing one or more of the following values:

LOG_CONS If **syslog**() cannot pass the message to syslogd(8) it will attempt to write the message to the console ("*/dev/console*").

LOG_NDELAY Open the connection to syslogd(8) immediately. Normally the open is delayed until the first message is logged. Useful for programs that need to manage the order in which file descriptors are allocated.

LOG_PERROR Write the message to standard error output as well to the system log.

LOG_PID Log the process id with each message: useful for identifying instantiations of daemons. On FreeBSD, this option is enabled by default and cannot be disabled.

The *facility* argument encodes a default facility to be assigned to all messages that do not have an explicit facility encoded:

LOG_AUTH The authorization system: login(1), su(1), getty(8), etc.

LOG_AUTHPRIV The same as LOG_AUTH, but logged to a file readable only by selected individuals.

LOG_CONSOLE Messages written to */dev/console* by the kernel console output driver.

LOG_CRON The cron daemon: cron(8).

LOG_DAEMON System daemons, such as routed(8), that are not provided for explicitly by other facilities.

LOG_FTP The file transfer protocol daemons: ftpd(8), tftpd(8).

LOG_KERN Messages generated by the kernel. These cannot be generated by any user processes.

LOG_LPR The line printer spooling system: lpr(1), lpc(8), lpd(8), etc.

LOG_MAIL The mail system.

LOG_NEWS The network news system.

LOG_NTP The network time protocol system.

LOG_SECURITY Security subsystems, such as ipfw(4).

LOG_SYSLOG Messages generated internally by syslogd(8).

LOG_USER Messages generated by random user processes. This is the default facility identifier if none is specified.

LOG_UUCP The uucp system.

LOG_LOCAL0 Reserved for local use. Similarly for LOG_LOCAL1 through LOG_LOCAL7.

The **closelog**() function can be used to close the log file.

The **setlogmask**() function sets the log priority mask to *maskpri* and returns the previous mask. Calls to **syslog**() with a priority not set in *maskpri* are rejected. The mask for an individual priority *pri* is calculated by the macro **LOG_MASK**(*pri*); the mask for all priorities up to and including *toppri* is given

by the macro **LOG_UPTO**(*toppri*);. The default allows all priorities to be logged.

## RETURN VALUES

The routines **closelog**(), **openlog**(), **syslog**() and **vsyslog**() return no value.

The routine **setlogmask**() always returns the previous log mask level.

## EXAMPLES

```
syslog(LOG_ALERT, "who: internal error 23");

openlog("ftpd", LOG_PID | LOG_NDELAY, LOG_FTP);

setlogmask(LOG_UPTO(LOG_ERR));

syslog(LOG_INFO, "Connection from host %d", CallingHost);

syslog(LOG_ERR|LOG_LOCAL2, "foobar error: %m");
```

## SEE ALSO

logger(1), syslogd(8)

## HISTORY

These functions appeared in 4.2BSD.

## BUGS

Never pass a string with user-supplied data as a format without using ‘%s’. An attacker can put format specifiers in the string to mangle your stack, leading to a possible security hole. This holds true even if the string was built using a function like **snprintf**(), as the resulting string may still contain user-supplied conversion specifiers for later interpolation by **syslog**().

Always use the proper secure idiom:

```
syslog(priority, "%s", string);
```