NAME

syslog, openlog, closelog, setlogmask – control system log

SYNOPSIS

#include <syslog.h>

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

void syslog(int *priority*, **char** **logstring*, **/*** *parameters* ***/ . . .);**

void closelog(void);

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

MT-LEVEL

Safe

DESCRIPTION

syslog( ) passes a message to **syslogd**(1M), which may append it to a log file, write it to the system console, or forward it (to either a list of users or **syslogd** on another host on the network), depending on the configuration of **/etc/syslog.conf**. *logstring* is tagged with a priority of *priority*, and looks like a **printf**(3B) string with one additional allowable format specification, **%m**, which is replaced with the error message string corresponding to the error number in **errno**. A trailing NEWLINE is added if needed. Options passed to **openlog( )** may cause the size of the message to expand. The maximum size of the message passed to **syslogd** is 1024 bytes.

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 bitwise inclusive OR of zero or more of the following flags, defined in the header **<syslog.h>**.

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, such as hard device errors.
LOG_ERR	Errors.
LOG_WARNING	Warning messages.
LOG_NOTICE	Conditions that are not error conditions, but that may require special handling.
LOG_INFO	Informational messages.
LOG_DEBUG	Messages that contain information normally of use only when debugging a program.

If special processing is needed, **openlog( )** can be called to initialize the log file. The parameter *ident* is a string that is prepended to every message. *logopt* is a bit field indicating logging options. Values for *logopt* are:

LOG_PID	Log the process ID with each message. This is useful for identifying specific daemon processes (for daemons that fork).

LOG_CONS	Write messages to the system console if they cannot be sent to **syslogd**(1M). This option is safe to use in daemon processes that have no controlling terminal, since **syslog( )** forks before opening the console.
LOG_NDELAY	Open the connection to **syslogd**(1M) immediately. Normally the open is delayed until the first message is logged. This is useful for programs that need to manage the order in which file descriptors are allocated.
LOG_NOWAIT	Do not wait for child processes that have been forked to log messages onto the console. This option should be used by processes that enable notification of child termination using **SIGCHLD**, since **syslog( )** may otherwise block waiting for a child whose exit status has already been collected.

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

LOG_KERN	Messages generated by the kernel. These cannot be generated by any user processes.
LOG_USER	Messages generated by random user processes. This is the default facility identifier if none is specified.
LOG_MAIL	The mail system.
LOG_DAEMON	System daemons, such as **in.ftpd**(1M).
LOG_AUTH	The authorization system: **login**(1), **su**(1M), **getty**(1M), etc.
LOG_LPR	The line printer spooling system: **lpr**(1B), **lpc**(1B), etc.
LOG_NEWS	Reserved for the USENET network news system.
LOG_UUCP	Reserved for the UUCP system; it does not currently use **syslog**.
LOG_CRON	The **cron/at** facility; **crontab**(1), **at**(1), **cron**(1M), etc.
LOG_LOCAL0	Reserved for local use.
LOG_LOCAL1	Reserved for local use.
LOG_LOCAL2	Reserved for local use.
LOG_LOCAL3	Reserved for local use.
LOG_LOCAL4	Reserved for local use.
LOG_LOCAL5	Reserved for local use.
LOG_LOCAL6	Reserved for local use.
LOG_LOCAL7	Reserved for local use.

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

setlogmask( ) 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.

EXAMPLES

This call logs a message at priority **LOG_ALERT**:

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

The FTP daemon **ftpd** would make this call to **openlog( )** to indicate that all messages it logs should have an identifying string of **ftpd**, should be treated by **syslogd**(1M) as other messages from system daemons are, should include the process ID of the process logging the message:

openlog("ftpd", LOG_PID, LOG_DAEMON);

Then it would make the following call to **setlogmask( )** to indicate that messages at priorities from **LOG_EMERG** through **LOG_ERR** should be logged, but that no messages at any other priority should be logged:

setlogmask(LOG_UPTO(LOG_ERR));

Then, to log a message at priority **LOG_INFO**, it would make the following call to **syslog**:

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

A locally-written utility could use the following call to **syslog( )** to log a message at priority **LOG_INFO** to be treated by **syslogd**(1M) as other messages to the facility **LOG_LOCAL2** are:

syslog(LOG_INFO | LOG_LOCAL2, "error: %m");

SEE ALSO

at(1), **crontab**(1), **logger**(1), **login**(1), **lpc**(1B), **lpr**(1B), **cron**(1M), **getty**(1M), **in.ftpd**(1M), **su**(1M), **syslogd**(1M), **printf**(3B), **syslog.conf**(4)