man pages section 3: Basic Library Functions

syslog(3C)

NAME

syslog, openlog, closelog, setlogmask- control system log

SYNOPSIS

```
#include <syslog.h>
```

void openlog(const char **ident*, int *logopt*, int *facility*);
void syslog(int *priority*, const char **message*, .../* *arguments* */);
void closelog(void);
int setlogmask(int *maskpri*);

DESCRIPTION

The **`syslog()`** function sends a message to syslogd(1M), which, depending on the configuration of `/etc/syslog.conf`, logs it in an appropriate system log, writes it to the system console, forwards it to a list of users, or forwards it to `syslogd` on another host over the network. The logged message includes a message header and a message body. The message header consists of a facility indicator, a severity level indicator, a timestamp, a tag string, and optionally the process ID.

The message body is generated from the *message* and following arguments in the same manner as if these were arguments to printf(3UCB), except that occurrences of `%m` in the format string pointed to by the *message* argument are replaced by the error message string associated with the current value of `errno`. A trailing NEWLINE character is added if needed.

Values of the *priority* argument are formed by ORing together a *severity level* value and an optional *facility* value. If no facility value is specified, the current default facility value is used.

Possible values of severity level include:

`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.

The facility indicates the application or system component generating the message. Possible facility values include:

`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).

`LOG_LPR`

The line printer spooling system: lpr(1B), lpc(1B).

`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).

`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.

The `openlog()` function sets process attributes that affect subsequent calls to `syslog()`. The *ident* argument is a string that is prepended to every message. The *logopt* argument indicates logging options. Values for *logopt* are constructed by a bitwise-inclusive OR of zero or more of the following:

`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_ODELAY`

Delay open until `syslog()` is called.

`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* argument encodes a default facility to be assigned to all messages that do not have an explicit facility already encoded. The initial default facility is `LOG_USER`.

The `openlog()` and `syslog()` functions may allocate a file descriptor. It is not necessary to call `openlog()` prior to calling `syslog()`.

The `closelog()` function closes any open file descriptors allocated by previous calls to `openlog()` or `syslog()`.

The `setlogmask()` function sets the log priority mask for the current process to *maskpri* and returns the previous mask. If the *maskpri* argument is 0, the current log mask is not modified. Calls by the current process 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_UPT(`*toppri*`)`. The default log mask allows all priorities to be logged.

Symbolic constants for use as values of the *logopt*, *facility*, *priority*, and *maskpri* arguments are defined in the <`syslog.h`> header.

RETURN VALUES

The `setlogmask()` function returns the previous log priority mask. The `closelog()`, `openlog()` and `syslog()` functions return no value.

ERRORS

No errors are defined.

EXAMPLES

Example 1 Example of `LOG_ALERT` message.

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");
```

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Safe

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(3UCB), syslog.conf(4), attributes(5)

SunOS 5.9 Last Revised 29 Dec 1996