35.13. `syslog` — Unix syslog library routines

This module provides an interface to the Unix `syslog` library routines. Refer to the Unix manual pages for a detailed description of the `syslog` facility.

This module wraps the system `syslog` family of routines. A pure Python library that can speak to a syslog server is available in the `logging.handlers` module as `SysLogHandler`.

The module defines the following functions:

`syslog.`**`syslog`**(*message*)
`syslog.`**`syslog`**(*priority*, *message*)
Send the string *message* to the system logger. A trailing newline is added if necessary. Each message is tagged with a priority composed of a *facility* and a *level*. The optional *priority* argument, which defaults to `LOG_INFO`, determines the message priority. If the facility is not encoded in *priority* using logical-or (`LOG_INFO | LOG_USER`), the value given in the `openlog()` call is used.

If `openlog()` has not been called prior to the call to `syslog()`, `openlog()` will be called with no arguments.

`syslog.`**`openlog`**([*ident*[, *logoption*[, *facility*]]])
Logging options of subsequent `syslog()` calls can be set by calling `openlog()`. `syslog()` will call `openlog()` with no arguments if the log is not currently open.

The optional *ident* keyword argument is a string which is prepended to every message, and defaults to `sys.argv[0]` with leading path components stripped. The optional *logoption* keyword argument (default is 0) is a bit field – see below for possible values to combine. The optional *facility* keyword argument (default is `LOG_USER`) sets the default facility for messages which do not have a facility explicitly encoded.

Changed in version 3.2: In previous versions, keyword arguments were not allowed, and *ident* was required. The default for *ident* was dependent on the system libraries, and often was `python` instead of the name of the python program file.

`syslog.`**`closelog`**()
Reset the syslog module values and call the system library `closelog()`.

This causes the module to behave as it does when initially imported. For example, `openlog()` will be called on the first `syslog()` call (if `openlog()` hasn't already been called), and *ident* and other `openlog()` parameters are reset to defaults.

`syslog.`**`setlogmask`**(*maskpri*)

Set the priority mask to *maskpri* and return the previous mask value. Calls to `syslog()` with a priority level not set in *maskpri* are ignored. The default is to log all priorities. The function `LOG_MASK(pri)` calculates the mask for the individual priority *pri*. The function `LOG_UPTO(pri)` calculates the mask for all priorities up to and including *pri*.

The module defines the following constants:

Priority levels (high to low):

`LOG_EMERG`, `LOG_ALERT`, `LOG_CRIT`, `LOG_ERR`, `LOG_WARNING`, `LOG_NOTICE`, `LOG_INFO`, `LOG_DEBUG`.

Facilities:

`LOG_KERN`, `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_LPR`, `LOG_NEWS`, `LOG_UUCP`, `LOG_CRON`, `LOG_SYSLOG`, `LOG_LOCAL0` to `LOG_LOCAL7`, and, if defined in `<syslog.h>`, `LOG_AUTHPRIV`.

Log options:

`LOG_PID`, `LOG_CONS`, `LOG_NDELAY`, and, if defined in `<syslog.h>`, `LOG_ODELAY`, `LOG_NOWAIT`, and `LOG_PERROR`.

35.13.1. Examples

35.13.1.1. Simple example

A simple set of examples:

```
import syslog

syslog.syslog('Processing started')
if error:
    syslog.syslog(syslog.LOG_ERR, 'Processing started')
```

An example of setting some log options, these would include the process ID in logged messages, and write the messages to the destination facility used for mail logging:

```
syslog.openlog(logoption=syslog.LOG_PID, facility=syslog.LOG_MAIL)
syslog.syslog('E-mail processing initiated...')
```