

```

...
33:
34: /*
35:  * Facility codes
36:  */
37:
38: #define LOG_KERN      (0<<3) /* kernel messages */
39: #define LOG_USER      (1<<3) /* random user-level messages */
40: #define LOG_MAIL      (2<<3) /* mail system */
41: #define LOG_DAEMON    (3<<3) /* system daemons */
42: #define LOG_AUTH      (4<<3) /* security/authorization messages */
43: #define LOG_SYSLOG    (5<<3) /* messages generated internally by syslogd */
44: #define LOG_LPR       (6<<3) /* line printer subsystem */
45: #define LOG_NEWS      (7<<3) /* netnews subsystem */
46: #define LOG_UUCP      (8<<3) /* uucp subsystem */
47: #define LOG_LFMT      (14<<3) /* logalert facility */
48: #define LOG_CRON      (15<<3) /* cron/at subsystem */
49: /* other codes through 15 reserved for system use */
50: #define LOG_LOCAL0    (16<<3) /* reserved for local use */
51: #define LOG_LOCAL1    (17<<3) /* reserved for local use */
52: #define LOG_LOCAL2    (18<<3) /* reserved for local use */
53: #define LOG_LOCAL3    (19<<3) /* reserved for local use */
54: #define LOG_LOCAL4    (20<<3) /* reserved for local use */
55: #define LOG_LOCAL5    (21<<3) /* reserved for local use */
56: #define LOG_LOCAL6    (22<<3) /* reserved for local use */
57: #define LOG_LOCAL7    (23<<3) /* reserved for local use */
58:
59: #define LOG_NFACILITIES 24 /* maximum number of facilities */
60: #define LOG_FACMASK    0x03f8 /* mask to extract facility part */
61:
62: /*
63:  * Priorities (these are ordered)
64:  */
65:
66: #define LOG_EMERG      0 /* system is unusable */
67: #define LOG_ALERT      1 /* action must be taken immediately */
68: #define LOG_CRIT       2 /* critical conditions */
69: #define LOG_ERR        3 /* error conditions */
70: #define LOG_WARNING    4 /* warning conditions */
71: #define LOG_NOTICE     5 /* normal but significant condition */
72: #define LOG_INFO       6 /* informational */
73: #define LOG_DEBUG      7 /* debug-level messages */
74:
75: #define LOG_PRIMASK    0x0007 /* mask to extract priority part (internal) */
76:
77: /*
78:  * arguments to setlogmask.
79:  */
80: #define LOG_MASK(pri)  (1 << (pri)) /* mask for one priority */
81: #define LOG_UPTO(pri)  ((1 << ((pri)+1)) - 1) /* all priorities through pri */
82:
83: /*
84:  * Option flags for openlog.
85:  */
86: * LOG_ODELAY no longer does anything; LOG_NDELAY is the
87: * inverse of what it used to be.
88: */
89: #define LOG_PID        0x01 /* log the pid with each message */
90: #define LOG_CONS       0x02 /* log on the console if errors in sending */
91: #define LOG_ODELAY     0x04 /* delay open until syslog() is called */
92: #define LOG_NDELAY     0x08 /* don't delay open */
93: #define LOG_NOWAIT     0x10 /* if forking to log on console, don't wait() */
94:
95: #if defined(__STDC__)
96: extern void syslog(int, const char *, ...);
97: extern void openlog(const char *, int, int);
98: extern void closelog(void);
99: extern int setlogmask(int);
100: #else
101: extern void syslog();
102: extern void openlog();
103: extern void closelog();
104: extern int setlogmask();
105: #endif
106:
107: #if defined(__cplusplus)
108: }
109: #endif

```

```

...
38:
39:
40: #define _PATH_LOG     "/dev/log"
41:
42: /*
43:  * priorities/facilities are encoded into a single 32-bit quantity, where the
44:  * bottom 3 bits are the priority (0-7) and the top 28 bits are the facility
45:  * (0-10-big number). Both the priorities and the facilities map roughly
46:  * one-to-one to strings in the syslogd(8) source code. This mapping is
47:  * included in this file.
48:  */
49: * priorities (these are ordered)
50: */
51: #define LOG_EMERG      0 /* system is unusable */
52: #define LOG_ALERT      1 /* action must be taken immediately */
53: #define LOG_CRIT       2 /* critical conditions */
54: #define LOG_ERR        3 /* error conditions */
55: #define LOG_WARNING    4 /* warning conditions */
56: #define LOG_NOTICE     5 /* normal but significant condition */
57: #define LOG_INFO       6 /* informational */
58: #define LOG_DEBUG      7 /* debug-level messages */
59:
60: #define LOG_PRIMASK    0x07 /* mask to extract priority part (internal) */
61: /* extract priority */
...
91:
92: /* facility codes */
93: #define LOG_KERN      (0<<3) /* kernel messages */
94: #define LOG_USER      (1<<3) /* random user-level messages */
95: #define LOG_MAIL      (2<<3) /* mail system */
96: #define LOG_DAEMON    (3<<3) /* system daemons */
97: #define LOG_AUTH      (4<<3) /* security/authorization messages */
98: #define LOG_SYSLOG    (5<<3) /* messages generated internally by syslogd */
99: #define LOG_LPR       (6<<3) /* line printer subsystem */
100: #define LOG_NEWS      (7<<3) /* network news subsystem */
101: #define LOG_UUCP      (8<<3) /* UUCP subsystem */
102: #define LOG_CRON      (9<<3) /* clock daemon */
103: #define LOG_AUTHPRIV  (10<<3) /* security/authorization messages (private) */
104: #define LOG_FTP        (11<<3) /* ftp daemon */
105:
106: /* other codes through 15 reserved for system use */
107: #define LOG_LOCAL0    (16<<3) /* reserved for local use */
108: #define LOG_LOCAL1    (17<<3) /* reserved for local use */
109: #define LOG_LOCAL2    (18<<3) /* reserved for local use */
110: #define LOG_LOCAL3    (19<<3) /* reserved for local use */
111: #define LOG_LOCAL4    (20<<3) /* reserved for local use */
112: #define LOG_LOCAL5    (21<<3) /* reserved for local use */
113: #define LOG_LOCAL6    (22<<3) /* reserved for local use */
114: #define LOG_LOCAL7    (23<<3) /* reserved for local use */
115:
116: #define LOG_NFACILITIES 24 /* current number of facilities */
117: #define LOG_FACMASK    0x03f8 /* mask to extract facility part */
118: /* facility of pri */
119: #define LOG_FAC(p)    (((p) & LOG_FACMASK) >> 3)
120:
...
149:
150: /*
151:  * arguments to setlogmask.
152:  */
153: #define LOG_MASK(pri)  (1 << (pri)) /* mask for one priority */
154: #define LOG_UPTO(pri)  ((1 << ((pri)+1)) - 1) /* all priorities through pri */
155:
156: /*
157:  * Option flags for openlog.
158:  */
159: * LOG_ODELAY no longer does anything.
160: * LOG_NDELAY is the inverse of what it used to be.
161: */
162: #define LOG_PID        0x01 /* log the pid with each message */
163: #define LOG_CONS       0x02 /* log on the console if errors in sending */
164: #define LOG_ODELAY     0x04 /* delay open until first syslog() (default) */
165: #define LOG_NDELAY     0x08 /* don't delay open */
166: #define LOG_NOWAIT     0x10 /* don't wait for console forks: DEPRECATED */
167: #define LOG_PERROR     0x20 /* log to stderr as well */
168:
169: __BEGIN_DECLS

```

```
110:
111: #endif /* _IO_SYSLOG_H */
```

```
170:
171: /* Close descriptor used to write to system logger. */
172: extern void closelog (void) __THROW;
173:
174: /* Open connection to system logger. */
175: extern void openlog (__const char *__ident, int __option, int __facility)
176:     __THROW;
177:
178: /* Set the log mask level. */
179: extern int setlogmask (int __mask) __THROW;
180:
181: /* Generate a log message using FMT string and option arguments. */
182: extern void syslog (int __pri, __const char *__fmt, ...) __THROW
183:     __attribute__((__format__(__printf__, 2, 3)));
184:
185: #ifdef __USE_BSD
186: /* Generate a log message using FMT and using arguments pointed to by AP. */
187: extern void vsyslog (int __pri, __const char *__fmt, __gnuc_va_list __ap)
188:     __THROW __attribute__((__format__(__printf__, 2, 0)));
189: #endif
190:
191: __END_DECLS
192:
193: #endif /* sys/syslog.h */
```