```
...
/*
 *  Facility codes
 */

#define LOG_KERN    (0<<3)    /* kernel messages */
#define LOG_USER    (1<<3)    /* random user-level messages */
#define LOG_MAIL    (2<<3)    /* mail system */
#define LOG_DAEMON    (3<<3)    /* system daemons */
#define LOG_AUTH    (4<<3)    /* security/authorization messages */
#define LOG_SYSLOG    (5<<3)    /* messages generated internally by syslogd */
#define LOG_LPR        (6<<3)    /* line printer subsystem */
#define LOG_NEWS    (7<<3)    /* netnews subsystem */
#define LOG_UUCP    (8<<3)    /* uucp subsystem */
#define LOG_LFMT    (14<<3) /* logalert facility */
#define    LOG_CRON    (15<<3)    /* cron/at subsystem */
    /* other codes through 15 reserved for system use */
#define LOG_LOCAL0    (16<<3)    /* reserved for local use */
#define LOG_LOCAL1    (17<<3)    /* reserved for local use */
#define LOG_LOCAL2    (18<<3)    /* reserved for local use */
#define LOG_LOCAL3    (19<<3)    /* reserved for local use */
#define LOG_LOCAL4    (20<<3)    /* reserved for local use */
#define LOG_LOCAL5    (21<<3)    /* reserved for local use */
#define LOG_LOCAL6    (22<<3)    /* reserved for local use */
#define LOG_LOCAL7    (23<<3)    /* reserved for local use */

#define LOG_NFACILITIES    24    /* maximum number of facilities */
#define LOG_FACMASK    0x03f8    /* mask to extract facility part */

/*
 *  Priorities (these are ordered)
 */

#define LOG_EMERG    0    /* system is unusable */
#define LOG_ALERT    1    /* action must be taken immediately */
#define LOG_CRIT    2    /* critical conditions */
#define LOG_ERR        3    /* error conditions */
#define LOG_WARNING    4    /* warning conditions */
#define LOG_NOTICE    5    /* normal but signification condition */
#define LOG_INFO    6    /* informational */
#define LOG_DEBUG    7    /* debug-level messages */

#define LOG_PRIMASK    0x0007    /* mask to extract priority part (internal) */

/*
 * arguments to setlogmask.
 */
#define    LOG_MASK(pri)    (1 << (pri))        /* mask for one priority */
#define    LOG_UPTO(pri)    ((1 << ((pri)+1)) - 1)    /* all priorities through pri */

/*
 *  Option flags for openlog.
 *
 *    LOG_ODELAY no longer does anything; LOG_NDELAY is the
 *    inverse of what it used to be.
 */
#define    LOG_PID        0x01    /* log the pid with each message */
#define    LOG_CONS    0x02    /* log on the console if errors in sending */
#define    LOG_ODELAY    0x04    /* delay open until syslog() is called */
#define LOG_NDELAY    0x08    /* don't delay open */
#define LOG_NOWAIT    0x10    /* if forking to log on console, don't wait() */

#if defined(__STDC__)
extern void syslog(int, const char *, ...);
extern void openlog(const char *, int, int);
extern void closelog(void);
extern int setlogmask(int);
#else
extern void syslog();
extern void openlog();
extern void closelog();
extern int setlogmask();
#endif

#if defined(__cplusplus)
    }
#endif
```

```
...


#define    _PATH_LOG    "/dev/log"

/*
 * priorities/facilities are encoded into a single 32-bit quantity, where the
 * bottom 3 bits are the priority (0-7) and the top 28 bits are the facility
 * (0-big number).  Both the priorities and the facilities map roughly
 * one-to-one to strings in the syslogd(8) source code.  This mapping is
 * included in this file.
 *
 * priorities (these are ordered)
 */
#define    LOG_EMERG    0    /* system is unusable */
#define    LOG_ALERT    1    /* action must be taken immediately */
#define    LOG_CRIT    2    /* critical conditions */
#define    LOG_ERR        3    /* error conditions */
#define    LOG_WARNING    4    /* warning conditions */
#define    LOG_NOTICE    5    /* normal but significant condition */
#define    LOG_INFO    6    /* informational */
#define    LOG_DEBUG    7    /* debug-level messages */

#define    LOG_PRIMASK    0x07    /* mask to extract priority part (internal) */
                /* extract priority */
...

/* facility codes */
#define    LOG_KERN    (0<<3)    /* kernel messages */
#define    LOG_USER    (1<<3)    /* random user-level messages */
#define    LOG_MAIL    (2<<3)    /* mail system */
#define    LOG_DAEMON    (3<<3)    /* system daemons */
#define    LOG_AUTH    (4<<3)    /* security/authorization messages */
#define    LOG_SYSLOG    (5<<3)    /* messages generated internally by syslogd */
#define    LOG_LPR        (6<<3)    /* line printer subsystem */
#define    LOG_NEWS    (7<<3)    /* network news subsystem */
#define    LOG_UUCP    (8<<3)    /* UUCP subsystem */
#define    LOG_CRON    (9<<3)    /* clock daemon */
#define    LOG_AUTHPRIV    (10<<3)    /* security/authorization messages (private) */
#define    LOG_FTP        (11<<3)    /* ftp daemon */

    /* other codes through 15 reserved for system use */
#define    LOG_LOCAL0    (16<<3)    /* reserved for local use */
#define    LOG_LOCAL1    (17<<3)    /* reserved for local use */
#define    LOG_LOCAL2    (18<<3)    /* reserved for local use */
#define    LOG_LOCAL3    (19<<3)    /* reserved for local use */
#define    LOG_LOCAL4    (20<<3)    /* reserved for local use */
#define    LOG_LOCAL5    (21<<3)    /* reserved for local use */
#define    LOG_LOCAL6    (22<<3)    /* reserved for local use */
#define    LOG_LOCAL7    (23<<3)    /* reserved for local use */

#define    LOG_NFACILITIES    24    /* current number of facilities */
#define    LOG_FACMASK    0x03f8    /* mask to extract facility part */
                /* facility of pri */
#define    LOG_FAC(p)    (((p) & LOG_FACMASK) >> 3)

...


/*
 * arguments to setlogmask.
 */
#define    LOG_MASK(pri)    (1 << (pri))        /* mask for one priority */
#define    LOG_UPTO(pri)    ((1 << ((pri)+1)) - 1)    /* all priorities through pri */

/*
 * Option flags for openlog.
 *
 * LOG_ODELAY no longer does anything.
 * LOG_NDELAY is the inverse of what it used to be.
 */
#define    LOG_PID        0x01    /* log the pid with each message */
#define    LOG_CONS    0x02    /* log on the console if errors in sending */
#define    LOG_ODELAY    0x04    /* delay open until first syslog() (default) */
#define    LOG_NDELAY    0x08    /* don't delay open */
#define    LOG_NOWAIT    0x10    /* don't wait for console forks: DEPRECATED */
#define    LOG_PERROR    0x20    /* log to stderr as well */

__BEGIN_DECLS
```


(For readability, not all matches are indicated.)

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

```
170:
171: /* Close desriptor 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 */
```


(For readability, not all matches are indicated.)